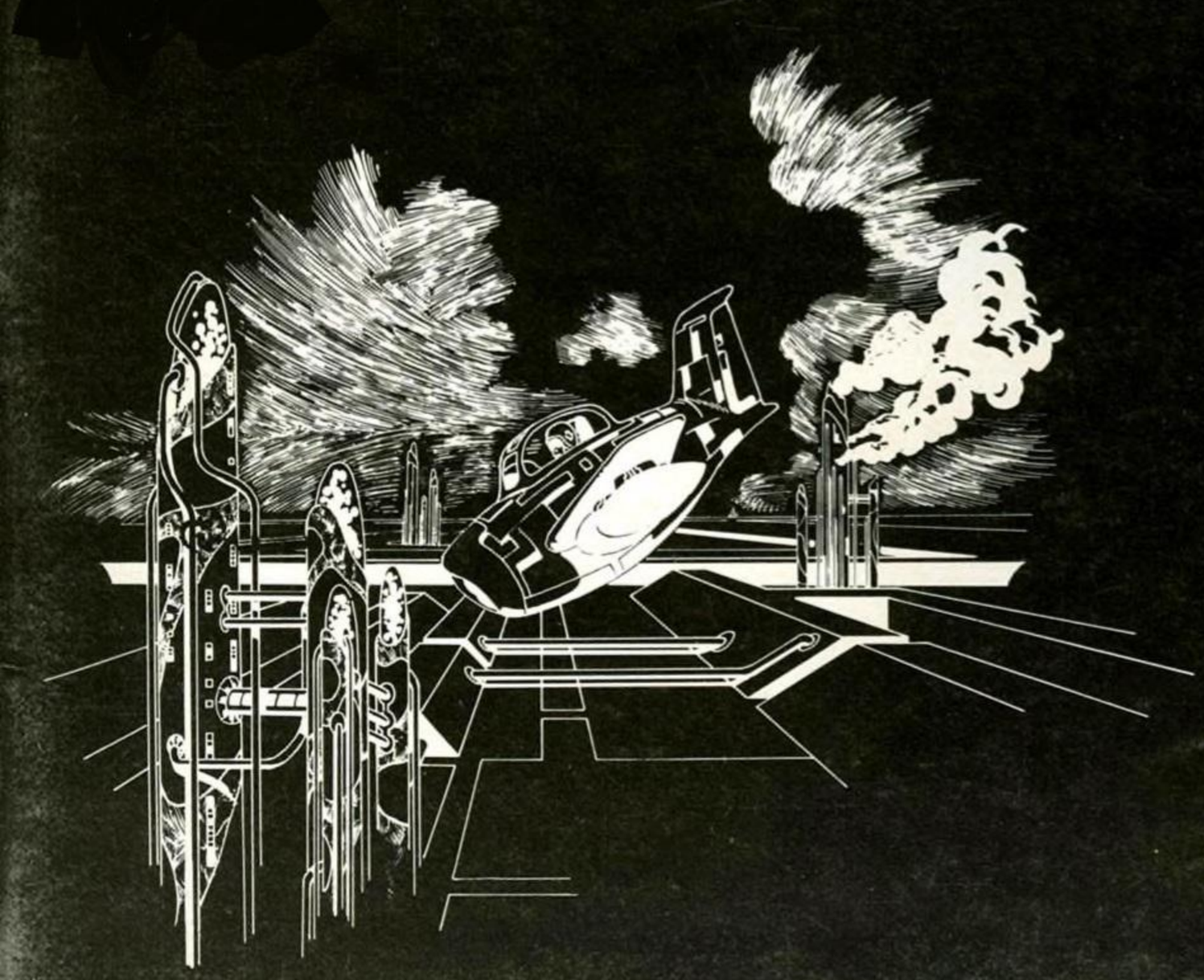
COMPUTER, PROGRAMMIERT ZUR UNTERHALTUNG

TI99 Collection Book

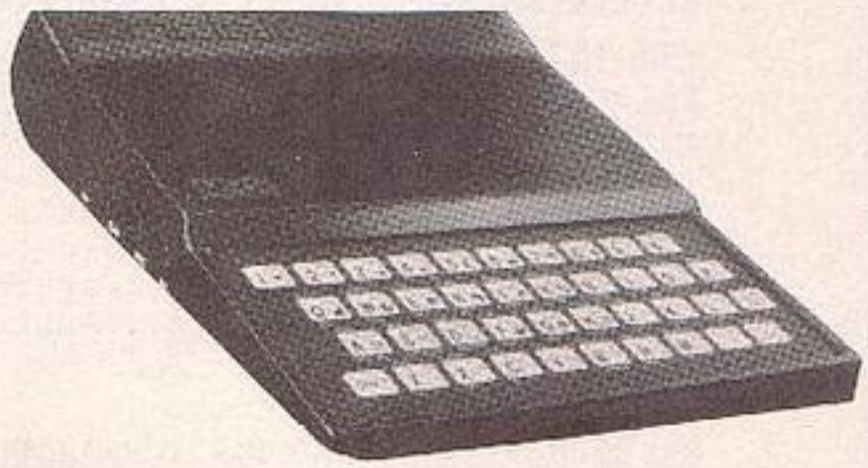
Software

Software Reviews

Programmier Tips



ZX-81: Wieviel Steckmodule möglich?



Herr Friedrich aus Köln fragt, ob es möglich ist, seinen mit einem 64K Memopak ausgestatteten ZX-81 durch Zuschalten eines zweiten Erweiterungsmoduls auf insgesamt 128KBYTE aufzustocken?
CPU: Die Idee ist nicht schlecht, aber leider aus zwei Gründen nicht zu realisieren. Erstens kann das System nur 64K verkraften, und zweitens ist es aus technischen Gründen nicht möglich, zwei Memopaks hintereinander an den ZX-81 zu adaptieren.
An ein aufgesetztes Memopak könnenjedoch weitere Peripheriegeräte wie z.B. Drucker oder auch ein I/O Bus zur Steuerung elektrischer Geräte, beispielweise Lichtquellen, angeschlossen werden.

TI 99/4A: Funktionstasten

TI 99/4A Anwender K.H. Wehner aus Hannover fragt, ob und wie die Funktionstasten seines Computers mit selbstdefinierten Zeichen belegt werden können. Das Handbuch bringt Ihn nicht weiter.
CPU: Aus dem Handbuch läßt sich nicht ohne weiteres die Anwendung der Funktionstasten erkennen.
Daher die folgenden Tips zur Programmierung der Funktionstasten und auch der Control-Tasten, die ebenfalls selbst belegt werden können.
Für die meisten Anwender und die Mehrzahl der Programme erweisen sich allerdings die Funktionstasten als das geeignete Instrument.
Im TI-Basic Modus bewirkt die Control-Taste in Verbindung mit den Tasten A-Z eine Änderung der Character Codes 65 - 90 hoch nach 128 - 153.
TI-Pascal benutzt die Control-Taste, um diesen Code von 1 nach 26 zu verschieben.
Die Funktionstaste in Verbindung mit den Tasten 1 - 9 ergibt folgende Werte:

Tasten	Hex-Code
FCTN und 1	03
FCTN und 2	04
FCTN und 3	07
FCTN und 4	02
FCTN und 5	0E
FCTN und 6	0C
FCTN und 7	01
FCTN und 8	06
FCTN und 9	0F

Diese Funktionen können beliebig eingesetzt werden, mit Ausnahme der Quit-Taste. Die bewirkt nämlich einen Abbruch im Programm.

Bilbo auf Wanderschaft

HOBBIT-Programm bald auch für andere Computer lieferbar

Wer kennt Ihn nicht, Bilbo Baggins, den Hobbit!

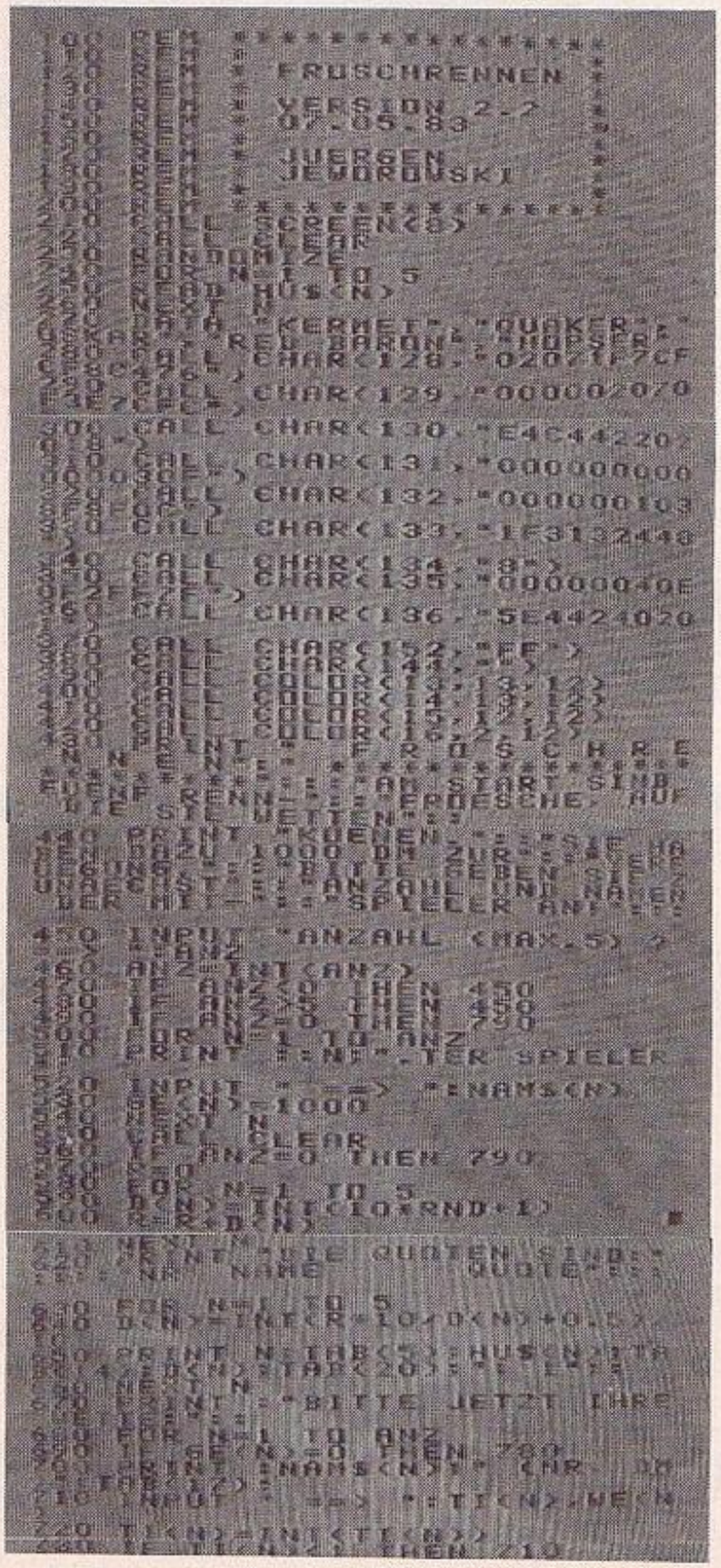
Nach dem Riesenerfolg den der Film und die Bücher 'Herr der Ringe' und 'Der kleine Hobbit'von J.R.R. Tolkien hatten, brachte Melbourne House Publishers, ein bekannter englischer Softwareanbieter, vor einigen Monaten das Abenteuerspiel 'The Hobbit' für den 48K Spectrum auf den Markt (s. Homecomputer 7/83 S.6). Innerhalb kurzer Zeit hat sich das Hobbit-Programm zu einem Riesenhit entwickelt und ist auch schon in der Bundesrepublik erhältlich, allerdings nur in der englischsprachigen Originalfassung.

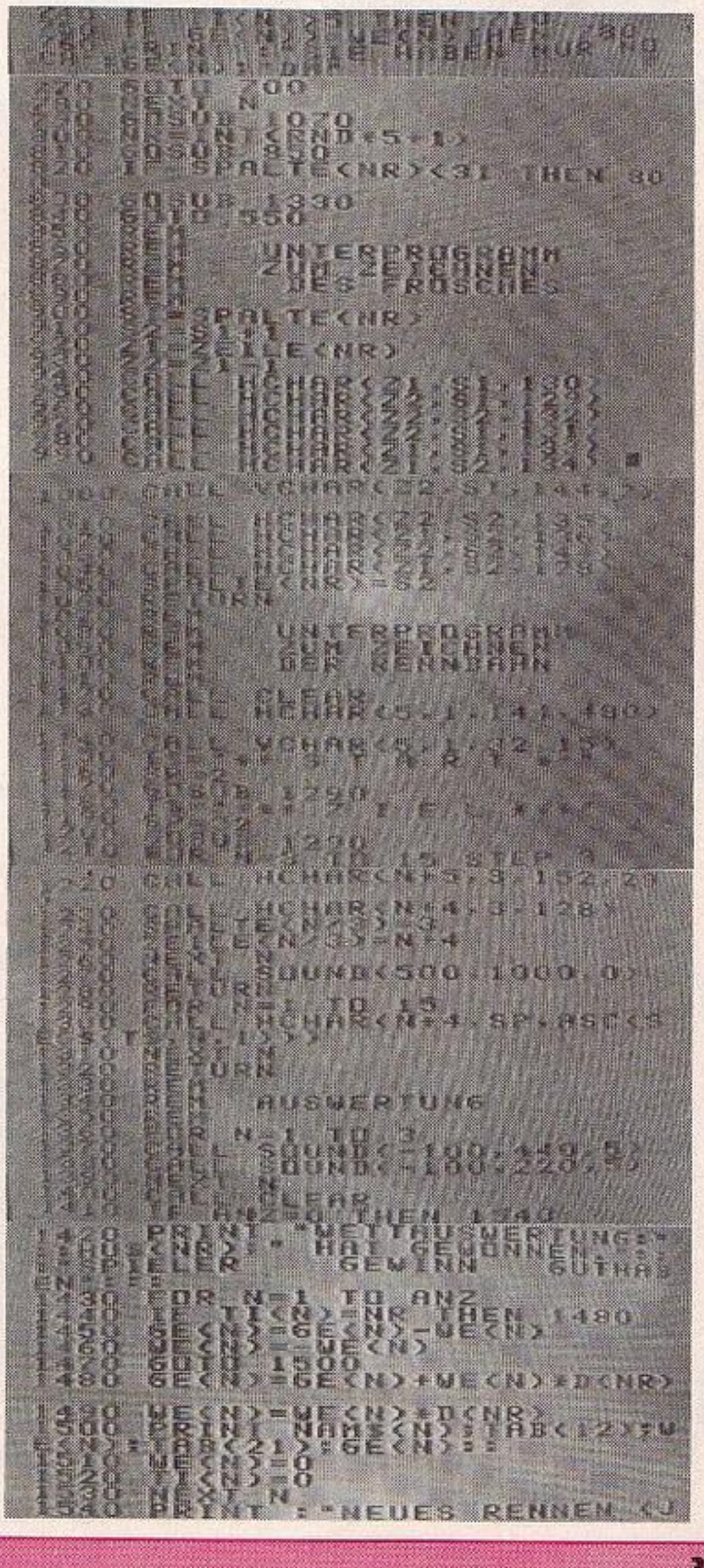
Seitdem wurde von Besitzern anderer Computersysteme ständig gefordert, dieses Programm für ihr eigenes Gerät umzuschreiben. Auch unsere Redaktion erhielt solche Anfragen.
Wir haben uns daher beim Hersteller erkundigt und dabei erfahren, daß man dort fieberhaft am arbeiten ist. Ab Herbst soll es das Hobbit-Spiel für drei weitere Computer geben: ORIC-1, BBC und Commodore 64. Wie bei der Spectrum-Version soll auch bei den neuen Hobbit-Programmen das Taschenbuch (in englischer Sprache natürlich) gleich mitgeliefert werden.
CPU wird zu gegebener Zeit weiter berichten.

Froschrennen

Ein lustiges Spiel für den TI-99/4A, bei dem fünf Frösche auf einer Rennstrecke um die Wette hüpfen.

Die Mitspieler können von ihrem Startkapital eine beliebige Summe auf ihren Favoriten setzen. Die Wettquoten, wie auch der Rennverlauf selber werden mit Hilfe von Zufallszahlen bestimmt.
Es können maximal fünf Spieler teilnehmen. Mehrfachwetten auf einen Frosch sind möglich. Wer sein ganzes Geld verspielt hat, scheidet aus.
Die Bewegung der Frösche wird wie beim Zeichentrickfilm aus mehreren Bildern zusammengesetzt und wirkt dadurch sehr fließend.
Nach dem Start des Programms müssen die Anzahl der Mitspieler und deren Namen eingegeben werden. Daraufhin erscheinen die für den folgenden Rennverlauf ermittelten Quoten. Jeder Spieler wird aufgefordert, die Nummer des Frosches, auf den er wetten will, sowie seinen Einsatz anzugeben. Dann wird das Rennen gestartet. Hat der erste Frosch das Ziel erreicht, so erfolgt die Gewinnausschüttung. Es können beliebig viele Rennen nacheinander durchgeführt werden. Hat ein Mitspieler sein gesamtes Kapital verspielt, so wird er im folgenden Rennablauf nicht mehr aufgeführt.
Die Bilder 1 bis 6 zeigen einen Probelauf.





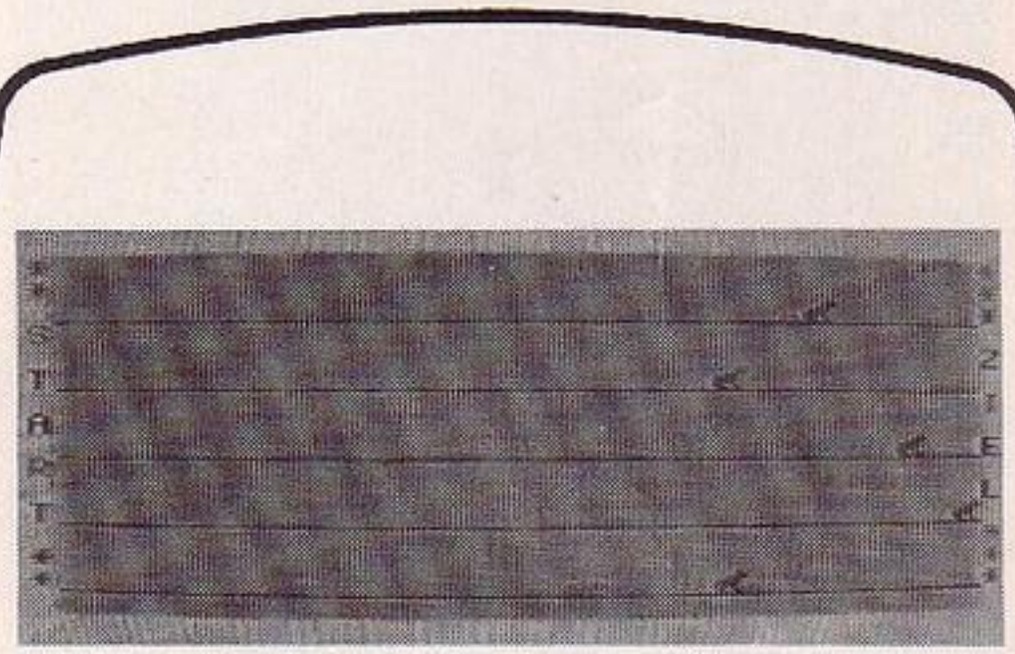
```
1500 CALL KEY(0,KEY,STATUS)
1510 IF STATUS=0 THEN 1550
1520 IF KEY<49 OR KEY>52 THEN 1600
1530 RETURN
```

```
1600 CALL SOUND(100,-2,0)
1610 GOTO 1550
1620 SKIP
1630 END
```

```
F R O S C H R E N N E N
*********************

AN START SIND FUENF RENN-
FROESCHE, AUF DIE SIE WETTEN
KOENEN.
SIE HABEN DAZU 1000 DM ZUR
VERFUEGUNG.
BITTE GEBEN SIE ZUNAECHST
ANZAHL UND NAMEN DER MIT-
SPIELER AN!

ANZAHL (MAX.5) ? <F> 4
```

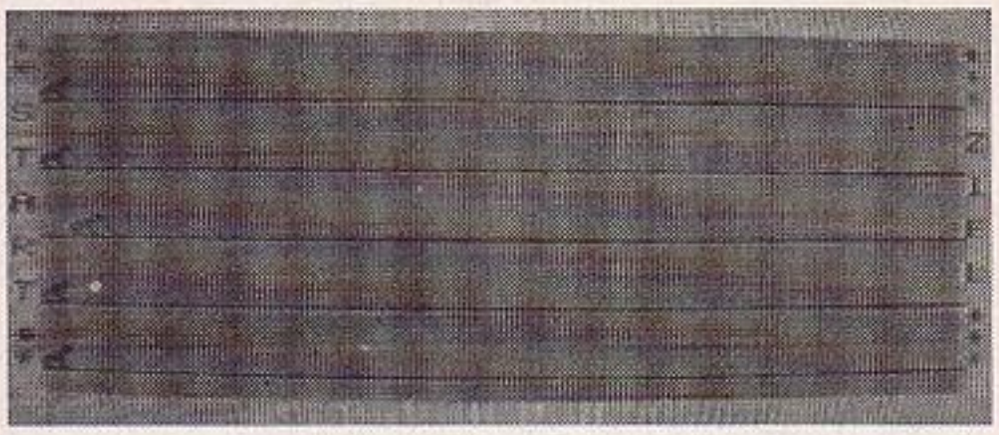


```
NR  NAME        QUOTE

 1  KERMIT       3.3   :  1
 2  QUAKER       5.8   :  1
 3  OSKAR        3.8   :  1
 4  RED BARON    5.3   :  1
 5  HOPSER       7.7   :  1

BITTE JETZT IHRE WETTE:

DORIS (NR, DM)     ==> 1,250
RALF (NR, DM)      ==> 4,100
ELKE (NR, DM)      ==> 2,50
JUERGEN (NR, DM)   ==> 3,100
```

```
WETTAUSWERTUNG:
RED BARON HAT GEWONNEN.

SPIELER      GEWINN     GUTHABEN

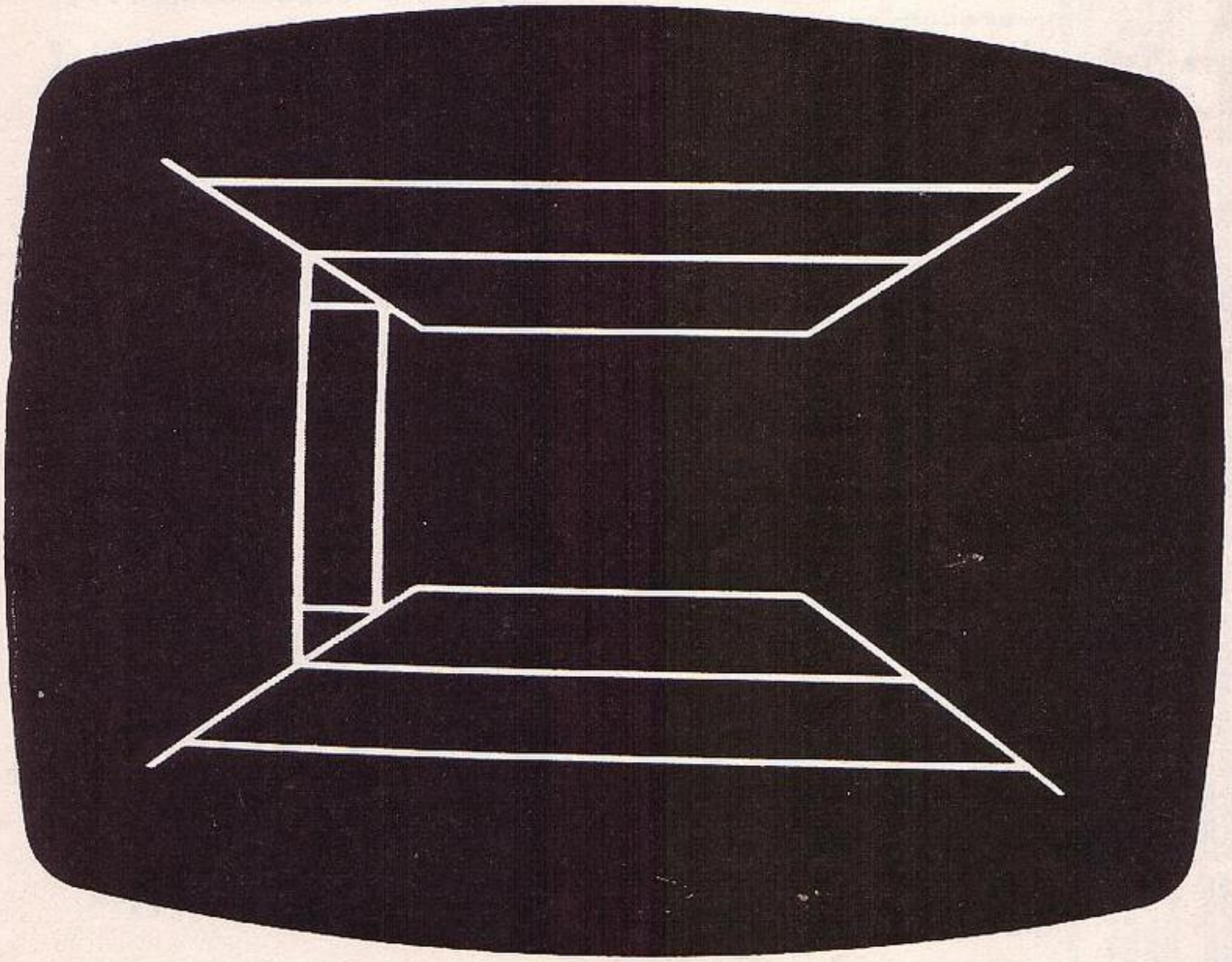
DORIS         -250         750
RALF           380        1580
ELKE           -50         950
JUERGEN        -100        900

NEUES RENNEN (J, N)
```

3-D-Labyrinth

Labyrinthe sind immer wieder faszinierend; besonderen Reiz erhalten sie aber bei einem Computer erst, wenn sie auf dem Bildschirm Dreidimensionalität simulieren. CPU stellt ein Labyrinth-Programm für den TI-99/4A vor.

Wie bei allen Labyrinthen, ist der Sinn des Spieles schnell erklärt: Sie befinden sich in einem Labyrinth! Finden Sie den Ausgang!
Über das Tastenfeld werden die einzelnen Schritte gesteuert.
Q - Drehung nach rechts
P - Drehung nach links
Space - ein Schritt vorwärts
A - Ausgabe des Labyrinths, was einem Mogeln gleichkommt.
Vorsicht: Kein Break!
Besitzer des MINI-MEMORY können die Grafik auf folgendem Wege noch einmal kräftig beschleunigen:
Statt der Zeilen 1291-1295:
1290 CALL POKEV(784,0,0,"",788,0,0,0,0,0,0,0,0)
Statt der Zeilen 1740-1743:
1740 CALL POKEV(785,0,"",788,0,0,0,0,0,0,0,0)
Und außerdem:
1770 CALL POKEV(785,240)-
1790 CALL POKEV(788,240)- Und die restlichen
1810 CALL POKEV(789,240)- CALL COLOR's entsprechend (1.Zahl:790-795 mit POKE 240 auffüllen.
Besitzer des EXTENDED BASIC gehen ähnlich vor: Auch sie verwenden bei 1290 und 1740 nur eine Anweisung, eben die erweiterte CALL COLOR - Anweisung des EX-BASIC (Das Setzen aller Farben auf einmal, ohne Schleife, bringt erheblichen Zeitgewinn).



```
10 REM   (C)MARC BRUENING
         JUNI 1983
         7-STUTTGART-75
      TEL.(0711)471367
100  CALL CLEAR
110  RANDOMIZE
120  PRINT "BITTE HABEN SIE E
TWA 1MINUTE6EDULD!":::::::::
::
130  DIM F(23,23),RT(3,1),U(2
,2)
140  DATA 0,1,0,-1,-1,0,1,0
150  FOR I=0 TO 1
160  FOR J=0 TO 3
170  READ RT(J,I)
180  NEXT J
190  NEXT I
200  FOR I=1 TO 23 STEP 2
210  FOR J=1 TO 23 STEP 2
220  F(I,J)=1
230  NEXT J
240  NEXT I
250  FOR I=0 TO 22 STEP
260  FOR J=1 TO 2
270  P=J+I-1
280  IF P<1 THEN 330
290  F(P,1)=J
300  F(P,2)=J
310  NEXT J
320  FOR I=3 TO 21 STEP 18
330  FOR J=1 TO 20
340  IF F(I,P)=2 THEN 378
350  F(I-11,P)=2
360  IF F(I-11,P)=1 THEN 390
370  IF P<23 THEN J
380  RETURN
390  FOR XH=1 TO 12
400  IF XH<24 THEN 450
410  XH=1
420  FOR YH=1 TO 23 STEP 2
430  IF F(XH,YH)=1 THEN 460
440  GOSUB 530
450  NEXT YH
460  NEXT XH
470  RETURN
480  REM PROGRAMMSTART
490  F(1,2)=-5
500  X=INT(RND*10)+3
510  Y=INT(RND*10)+3
520  IF F(X,Y)<>0 THEN 500
530  R=INT(RND*4)
540  GOSUB 1090
550  GOSUB 1740
560  CALL KEY(0,K,S)
570  IF S=0 THEN 770
580  IF K=81 THEN 840
590  IF K=83 THEN 880
600  IF K=65 THEN 980
610  GOTO 770
620  R=R-1
630  IF R>-1 THEN 750
640  R=3
650  GOTO 750
660  R=R+1
670  IF R<4 THEN 750
680  R=0
690  GOTO 750
700  X=X+RT(R,0)
710  Y=Y+RT(R,1)
720  IF F(X,Y)=1 THEN 770
730  X=UX
740  Y=UY
750  GOTO 750
760  CALL CLEAR
770  FOR I=1 TO 23
780  FOR J=1 TO 23
790  IF F(I,J)<1 THEN 1030
800  CALL HCHAR(J,I,58)
810  NEXT J
820  NEXT I
830  CALL KEY(0,K,S)
840  IF S=0 THEN 1050
850  GOSUB 1090
860  GOTO 790
870  CALL CLEAR
880  IF ST=1 THEN 1280
890  RESTORE 1390
900  CALL CHAR(58,"FFFFFFFFFF
FFFFFFFF")
910  FOR K=1 TO 10
920  READ A$
930  FOR I=1 TO 3
940  READ J
950  CALL CHAR(J,A$)
960  NEXT I
970  NEXT K
980  FOR K=1 TO 4
990  READ A$
1000 FOR I=1 TO 4
1010 READ J
1020 CALL CHAR(J,A$)
1030 NEXT I
1040 NEXT K
1050 CALL SCREEN(5)
1060 CALL COLOR(1,16,1)
1070 CALL COLOR(2,1,1)
1080 FOR I=5 TO 12
1090 CALL COLOR(I,1,1)
1100 NEXT I
1110 PRINT "                "
$$$$$$$$$$$$$$$$
1120 PRINT "                "
$$$$$$$$$$$$$$$$
1130 PRINT "                "
$$$$$$$$$$$$$$$$
1140 PRINT "                "
$$$$$$$$$$$$$$$$
1150 PRINT "                "
$$$$$$$$$$$$$$$$
1160 NEXT I
1170 NEXT K
1180 RETURN
1190 DATA 0101010101010101,6
5,88,112,8080808080808080,72,
96,120,FF,66,89,113,FF,73,97
,121
1200 DATA FF0101010101010101,6
7,90,114,FF8080808080808080,74,
98,122,0101010101010101FF,68,9
9,123,0000000000000000FF,69,
100,124
1210 DATA 8080808080808080FF,7
0,92,116,0000000000000000FF,76,
101,125
1220 DATA 8040201008040201,34
,40,80,104,0102040810204080,
35,41,81,105,0000000000000000FF
1230 DATA 36,42,82,106,FF,37
,43,83,107
1240 DATA 804020100804020,38,
44,84,108,0080402010080402,39,
45,85,109,0000000000000000
1250 NEXT I
1260 NEXT K
1270 NS=1
1280 CALL SCREEN(5)
1290 CALL COLOR(1,16,1)
1291 CALL COLOR(2,1,1)
1292 FOR I=5 TO 12
1293 CALL COLOR(I,1,1)
1294 CALL COLOR(I,1,1)
1295 NEXT I
1300 PRINT "  "$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$
1310 PRINT "  "*********
********
1320 PRINT "  "
1330 PRINT "  "
1340 PRINT "  "
1350 PRINT "  "
1360 PRINT "  "
1370 PRINT "  "
1380 RETURN
1390 DATA 0101010101010101,6
5,88,112,8080808080808080,72,
96,120,FF,66,89,113,FF,73,97
,121
1400 DATA FF0101010101010101,6
7,90,114,FF8080808080808080,74,
98,122,0101010101010101FF,68,9
9,123,0000000000000000FF,69,
100,124

1420 DATA 8040201008040201,3
4,40,80,104,0102040810204080,
35,41,81,105,0000000000000000F
F
1430 DATA 36,42,82,106,FF,37
,43,83,107
1440 ON R+1 GOSUB 1460,1530,
1600,1670
1450 RETURN
1460 FOR XI=0 TO 2
1470 FOR YI=0 TO 2
1480 IF Y-3+YI<1 THEN 1500
1490 U(XI,YI)=F(X-1+XI,Y-3+Y
I)
1500 NEXT YI
1510 NEXT XI
1520 RETURN
1530 FOR XI=0 TO 2
1540 FOR YI=0 TO 2
1550 IF X+3-YI>23 THEN 1570
1560 U(XI,YI)=F(X+3-YI,Y-1+X
I)
1570 NEXT YI
1580 NEXT XI
1590 RETURN
1600 FOR XI=0 TO 2
1610 FOR YI=0 TO 2
1620 IF Y+3-YI>23 THEN 1640
1630 U(XI,YI)=F(X+1-XI,Y+3-Y
I)
1640 NEXT YI
1650 NEXT XI
1660 RETURN
1670 FOR XI=0 TO 2
1680 FOR YI=0 TO 2
1690 IF X-3+YI<1 THEN 1710
1700 U(XI,YI)=F(X-3+YI,Y+1-X
I)
1710 NEXT YI
1720 NEXT XI
1730 RETURN
1740 CALL COLOR(2,1,1)
1741 FOR I=5 TO 12
1742 CALL COLOR(I,1,1)
1743 NEXT I
1750 IF U(1,2)<0 THEN 1970
1760 IF U(1,2)>0 THEN 1960
1770 CALL COLOR(2,16,1)
1780 IF U(0,2)>0 THEN 1800
1790 CALL COLOR(5,16,1)
1800 IF U(2,2)>0 THEN 1820
1810 CALL COLOR(6,16,1)
1820 IF U(1,1)<0 THEN 1970
1830 IF U(1,1)>0 THEN 1960
1840 CALL COLOR(7,16,1)
1850 IF U(0,1)>0 THEN 1870
1860 CALL COLOR(8,16,1)
1870 IF U(2,1)>0 THEN 1890
1880 CALL COLOR(9,16,1)
1890 IF U(1,0)<0 THEN 1970
1900 IF U(1,0)>0 THEN 1960
1910 CALL COLOR(10,16,1)
1920 IF U(0,0)>0 THEN 1940
1930 CALL COLOR(11,16,1)
1940 IF U(2,0)>0 THEN 1960
1950 CALL COLOR(12,16,1)
1960 RETURN
1970 END
```

Laser

für den TI-99/4A

Zum Bildschirm: Rechts ist der Tank. Sie bedienen die Lafette am unteren Rand mit
S **für links und**
D **für rechts.**
Mit *SPACE* feuern Sie einen Laser-Strahl ab.
Der Auftrag lautet: Die Felsen am oberen Rand zerstören! Wenn sich der Tank leert, wird allerdings auch der Strahl schwächer!

Jetzt die Schwierigkeit! Sie sind von den Felsen durch ein Feld von Spiegeln getrennt. Diese Spiegel reflektieren die Strahlen auf ganz natürliche Weise und werden von ihnen nach dem Auftreffen auf ganz unnatürliche Weise um 90 Grad gedreht! Zur Eingabe: Der Schwierigkeitsgrad reguliert die Größe der Felsen und des Spiegelfeldes. "Geschwindigkeit" setzt eine Warteschleife (d.h. 1 — schnell) ohne eigentlich das Spiel zu beeinflussen. Bei einer langsameren Geschwindigkeit ist die Grafik besser zu beobachten. Versuchen Sie am Anfang am besten mal 100.
Zur Grundidee: (Laser & Spiegel). Diese Grundidee läßt sich natürlich auf mannigfaltige Art abwandeln, ob zu Aktionsspielen für zwei Spieler, wie Mensch gegen Mensch oder Mensch gegen Computer. Oder noch besser als richtiges gutes Denkspiel.

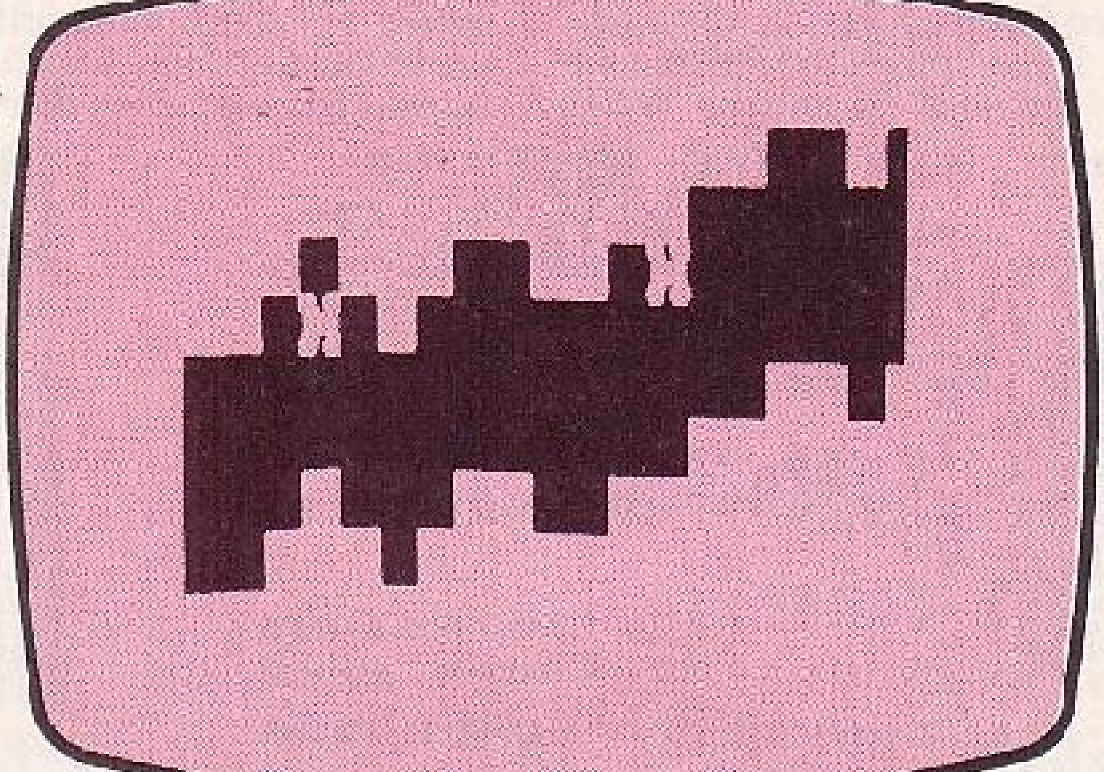
```
10 REM  (C) MARC BRUENING
        24.7.1983
        7-STUTTGART-75
        TEL.(0711)471367
90 INPUT "SCHWIERIGKEITSGRAD
?(1-6)":S6
91 INPUT "GESCHWINDIGKEIT?(1
-1000)":GS
92 RANDOMIZE
93 FOR I=1 TO S6
94 READ P
95 NEXT I
96 DATA 10,27,52,85,126,175
100 REM  LASER
101 T$="000000000000000FFFF
FFFFFFFFFFF"
102 T1=138
103 T2=17
110 CALL CLEAR
111 CALL CHAR(130,"55AA55AA5
5AA55AA")
112 CALL CHAR(128,"181818187
EFFC342")
120 CALL CHAR(131,"804020100
8040201")
130 CALL CHAR(129,"010204081
0204080")
140 CALL CHAR(132,"000000FF"
)
150 CALL CHAR(133,"101010101
0101010")
160 CALL CHAR(134,"80402010
8142418")
170 CALL CHAR(135,"9050301F0
8040201")
180 CALL CHAR(136,"010204081
F284888")
190 CALL CHAR(137,"090A0C0F8
1204080")
200 FOR I=138 TO 159
210 CALL CHAR(I,"FFFFFFFFFFF
FFFFF")
220 CALL HCHAR(I-136,29,I,3)
230 NEXT I
240 CALL HCHAR(1,1,30,32)
250 CALL HCHAR(24,1,30,32)
260 CALL VCHAR(2,1,30,22)
270 CALL VCHAR(2,32,30,22)
280 CALL VCHAR(2,28,30,22)
290 FOR I=12-S6/2 TO 12+S6/2
300 FOR J=2 TO 27
310 CALL HCHAR(I,J,130+INT(R
ND*2)*2-1)
320 NEXT J
330 NEXT I
340 FOR I=2 TO S6+2
350 FOR J=14-S6*2 TO 14+S6*2
360 CALL HCHAR(I,J,130)
370 NEXT J
380 NEXT I
390 Y=23
400 X=14
410 CALL HCHAR(Y,X,128)
415 IF SC=PEEK(0,K,S) THEN 10000
420 IF S=0 THEN 420
430 IF K=32 THEN 1000
450 IF K=83 THEN 500
460 IF K=68 THEN 600
470 GOTO 420
500 CALL HCHAR(Y,X,32)
510 X=X-1
520 IF X>1 THEN 700
530 X=2
540 GOTO 410
600 CALL HCHAR(Y,X,32)
610 X=X+1
620 IF X<28 THEN 700
630 X=27
640 GOTO 410
700 REM  TANK
710 IF T2=1 THEN 800
720 T2=T2-2
730 CALL CHAR(T1,SEG$(T$,T2,
16))
740 GOTO 410
800 IF T2=159 THEN 10000
810 T2=15
820 T1=T1+1
830 GOTO 730
1000 R=1
1001 G=0
1010 XS=X
1020 YS=22
1030 FOR I=1 TO 1000
1040 RUN "DSK1.LASER"
1050 IF G>((159-T1)*8+(T2-1)
/2)/32 THEN 1100
1060 NEXT I
1100 FOR I=1 TO 3
1110 CALL SOUND(10,110,1)
1120 NEXT I
1125 SC=SC+6-INT(6/10)*10
1130 GOTO 700
2000 YS=YS-1
2010 CALL GCHAR(YS,XS,C)
2020 IF C=32 THEN 2100
2030 IF C=129 THEN 2200
2040 IF C=131 THEN 2300
2050 IF C=130 THEN 2400
2060 IF C=128 THEN 10000
2070 G=10+G
2080 RETURN
2100 CALL HCHAR(YS,XS,133)
2105 GOSUB 6000
2110 CALL HCHAR(YS,XS,32)
2120 RETURN
2200 CALL HCHAR(YS,XS,136)
2205 R=2
2210 GOSUB 6000
2220 CALL HCHAR(YS,XS,131)
2230 RETURN
2300 CALL HCHAR(YS,XS,134)
2305 R=4
2310 GOSUB 6000
2320 CALL HCHAR(YS,XS,129)
2330 RETURN
2400 G=G+1
2410 GOTO 2100
3000 XS=XS+1
3010 CALL GCHAR(YS,XS,C)
3020 IF C=32 THEN 3100
3030 IF C=129 THEN 3200
3040 IF C=131 THEN 3300
3050 IF C=130 THEN 3400
3060 IF T=2060 THEN 2060
3100 CALL HCHAR(YS,XS,132)
3110 GOTO 2105
3200 CALL HCHAR(YS,XS,137)
3205 R=1
3210 GOTO 2210
3300 CALL HCHAR(YS,XS,134)
3305 R=3
3310 GOTO 2310
3400 G=G+1
4000 YS=YS+1
4010 CALL GCHAR(YS,XS,C)
4020 IF C=32 THEN 2100
4030 IF C=129 THEN 4200
4040 IF C=131 THEN 4300
4050 IF C=130 THEN 2400
4060 IF T=4060 THEN 4060
4200 CALL HCHAR(YS,XS,137)
4210 GOTO 2210
4300 CALL HCHAR(YS,XS,135)
4310 GOTO 2310
5000 XS=XS-1
5010 CALL GCHAR(YS,XS,C)
5030 IF C=129 THEN 5200
LIST 5040-
5040 IF C=131 THEN 5300
5050 IF C=130 THEN 3400
5060 GOTO 2060
5200 CALL HCHAR(YS,XS,136)
5205 R=3
5210 GOTO 2210
5300 CALL HCHAR(YS,XS,135)
5305 R=1
5310 GOTO 2310
6000 FOR J=1 TO 6S
6010 NEXT J
6020 RETURN
10000 CALL CLEAR
10010 PRINT SC;"POINTS":::::
::::::
10020 END
```

Nacht-Fahrer

für den TI-99/4A + Mini-Memory-Modul

Nachtfahrer ist ein einfaches Autofahrspiel. Die kleine Besonderheit ist der Maschinen-Scroll-Down, der mit den CALL-LOAD-Anweisungen in den Middle-Memory gepoket wird. Dieser Scroll-Down kann auf die gleiche Weise in andere Programme eingebunden werden. Er steht unter dem Namen SCROLL zur Verfügung. Aber Achtung! Er verwendet 768 Byte des RAM's als Zwischenspeicher!

Nun aber zum Spiel selbst: Sie steuern das helle Auto am unteren Rand
Z = links;
"Punkt/größer als" = rechts:
auf einer kurvenreichen Straße!
Am Anfang selten, später dann gehäuft, parken Rowdys die Straße zu. Weichen Sie aus!
Alle 500 Punkte = 'Schritte' = 1 Level wird die Straße erstmal wieder freier, aber nur um bis zum nächsten Mal noch voller zu werden. Wenn Sie in den Graben fahren oder ein parkendes Auto rammen, dann wird es recht ungemütlich !!!
Noch ein Tip: Wer sich eine Weile darin geübt hat, sollte den Schwierigkeitsgrad vom Programm schneller erhöhen lassen. Das verkürzt die Spielzeit und vermindert die Wahrscheinlichkeit, daß Sie vor Ihrer Könnensschranke aus dem Rennen geworfen werden: Es treten nämlich auch von Zeit zu Zeit unpassierbare Stellen auf!
Nun aber: Startschuß ..und ..LOS !!!



```
2 REM   (C) MARC BRUENING
            APRIL 1983
            7-STUTTGART-75
        TEL.(711)471367
3 REM  FUER KONFIGURATION:
TI99/4A&MINI MEMORY-MODUL
5 CALL CLEAR
6 DEF E=P/1000-F
10 PRINT "                NACHTFAH
RER"::::::::::::
11 FOR I=2 TO 15
12 CALL COLOR(I,16,2)
13 NEXT I
14 CALL SCREEN(2)
15 CALL COLOR(16,15,2)
16 CALL COLOR(1,5,5)
20 CALL INIT
21 CALL LOAD(32000,194,139,2
,0,0,2,1,113,24,2,2,2,224,
4,32,96,48,2,0,0,32,4,32,96,
40)
22 CALL LOAD(32026,2,0,0,0,2
,1,125,44,2,2,0,32,4,32,96,4
0,4,90) ■

23 FOR I=32044 TO 32076
24 CALL LOAD(I,128)
25 NEXT I
26 CALL LOAD(28702,127,224,"
",32736,83,67,82,79,76,76,12
5,0)
90 RANDOMIZE
100 CALL CHAR(132,"")
120 CALL CLEAR
130 CALL CHAR(155,"C3FFFF241
8DBFFC3")
150 CALL CHAR(130,"3C2424242
424243C")
160 CALL CHAR(131,"C3FFDB182
4FFFFC3")
170 PS=15
180 PA=16
183 CALL VCHAR(1,15,31,96)
190 CALL HCHAR(1,PS,31,4)
191 IF RND>E THEN 200
192 CALL POKEV(PS+INT(RND*3)
,251)
200 CALL PEEKV(736+PA,C)
210 IF C<>127 THEN 1000 ■

220 CALL POKEV(736+PA,227)
230 PS=PS+INT(RND*3)-1
240 IF PS>0 THEN 270
250 PS=1
270 IF PS<30 THEN 290
280 PS=29
290 CALL KEY(5,X,S)
300 IF S=0 THEN 500
310 IF X<>90 THEN 330
320 PA=PA-1
321 GOTO 400
330 IF X<>46 THEN 500
340 PA=PA+1
400 IF PA>-1 THEN 420
410 PA=0
420 IF PA<32 THEN 500
430 PA=32
500 CALL LINK("SCROLL")
505 P=P+1
506 IF P/500=INT(P/500)THEN
520
510 GOTO 190
520 F=F+.49
521 FOR I=-3 TO -1
522 CALL SOUND(-100,I,1)
523 NEXT I
530 GOTO 190
1000 RESTORE 1080
1010 CALL POKEV(736+PA,228)
1020 FOR I=1 TO 6
1030 CALL SOUND(-1000,-7,19-
3*I)
1040 READ A$
1050 CALL CHAR(132,A$)
1060 FOR J=1 TO 30
1070 NEXT J
1075 NEXT I
1080 DATA 0000001818,00003C3
C3C3C,9152381FF12C4A89,917E4
242C3427E89,FF81A18993C191FF
,0
1090 FOR I=1 TO 24
1100 CALL LINK("SCROLL")
1110 NEXT I
1120 PRINT "PUNKTZAHL:":P:"D
IES BEDEUTET LEVEL":INT(P/50
0)+1::
>■

 1130 FOR I=1 TO INT(P/500)+1
 1140 PRINT "*#*";
 1150 NEXT I
 1160 PRINT ::::::::::
 1170 END
>NACHTFAHRER:EIN SPIEL FUER
 DEN TI99/4A(KONSOLE&MODUL
 MINI-MEMORY!)
>EIN KLEINES MASCHINENPRO-
 GRAMM FUER EINEN SCROLL-DOWN
 WIRD IN DEN MIDDLE MEMORY
 GEPOKET
>TASTE Z:LINKS TASTE .:RECHTS


>VIEL SPASS!

>CALL HCHAR(21,3,32,30)
>■
```

Irrgarten für Katz und Maus

für den TI-99/4A

"Irrgarten für Katz und Maus", ein Spiel, für den TI-99/4A programmiert, mit dem Sie wiedereinmal Ihre Geschicklichkeit und Ihre Konzentrationsfähigkeit unter Beweis stellen können. Sollte es nicht gleich beim erstenmal gelingen, den gefräßigen Katzen zu entkommen, dann heißt es üben, üben und nochmals üben!!!

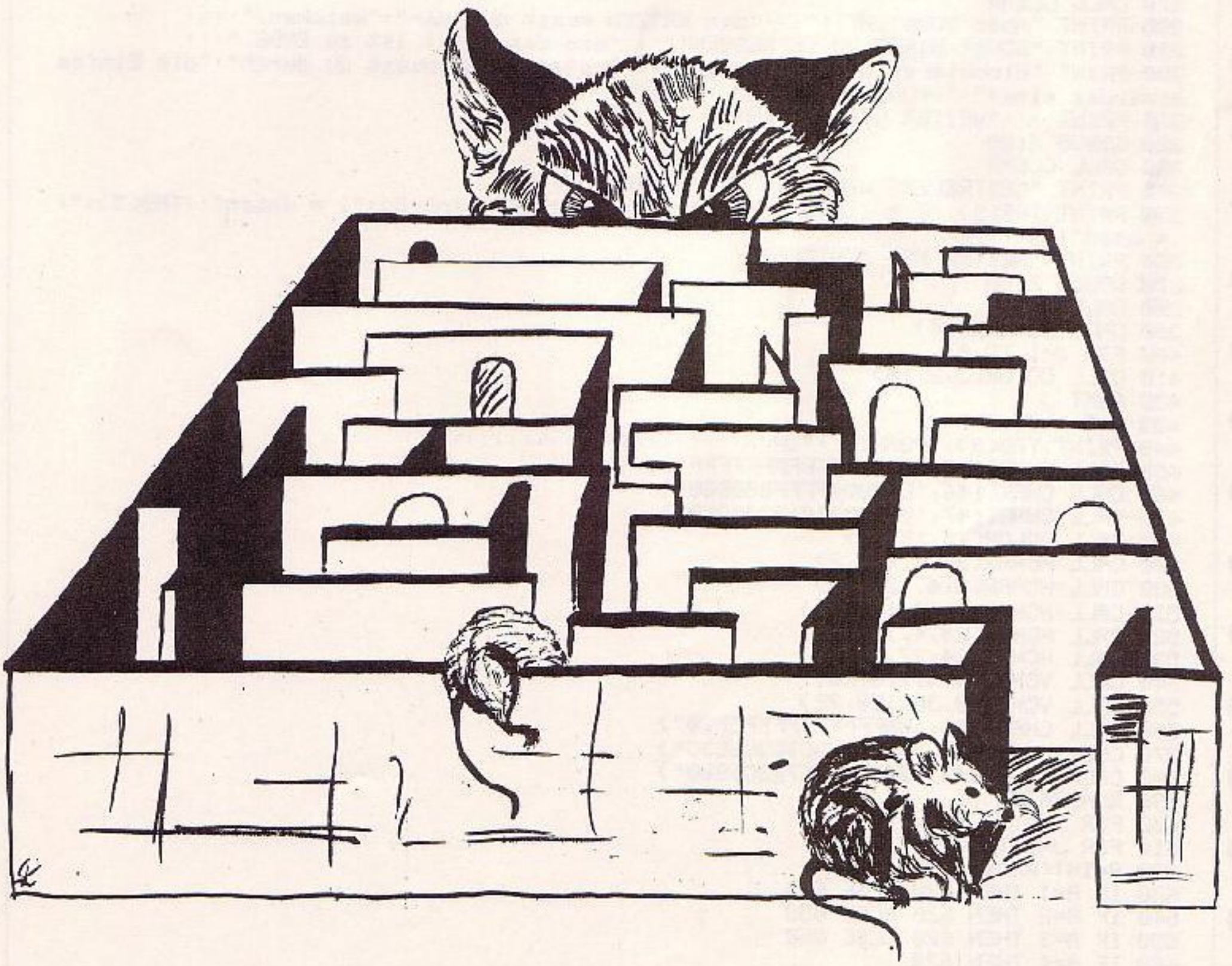
Der Sinn des Spieles liegt darin, die Maus von einem Ausgang zum anderen Ausgang zu führen. Man hat 5 Punkte als Bonus zur Verfügung. jeder Anstoß an die Einfassung kostet einen Punkt. Beim Verlassen der äusseren Einfriedung, wegen körperlicher Bedrängnis von den Katzen ausgehend, die Ihnen ja ans Leben wollen, ist das Spiel zu Ende.

Die Katzen lauern überall auf Sie, vor allen Dingen natürlich an den Ausgängen.

Falls die Katzen alle Ausgänge blockieren, muß man die Einfassung mit Punktverlust durchbrechen.

Das Spiel wird über die Tastatur gespielt und zwar mit folgenden Tasten:

Z = nach links
X = nach rechts
"größer als" = nach unten
"Semikolon" = nach oben



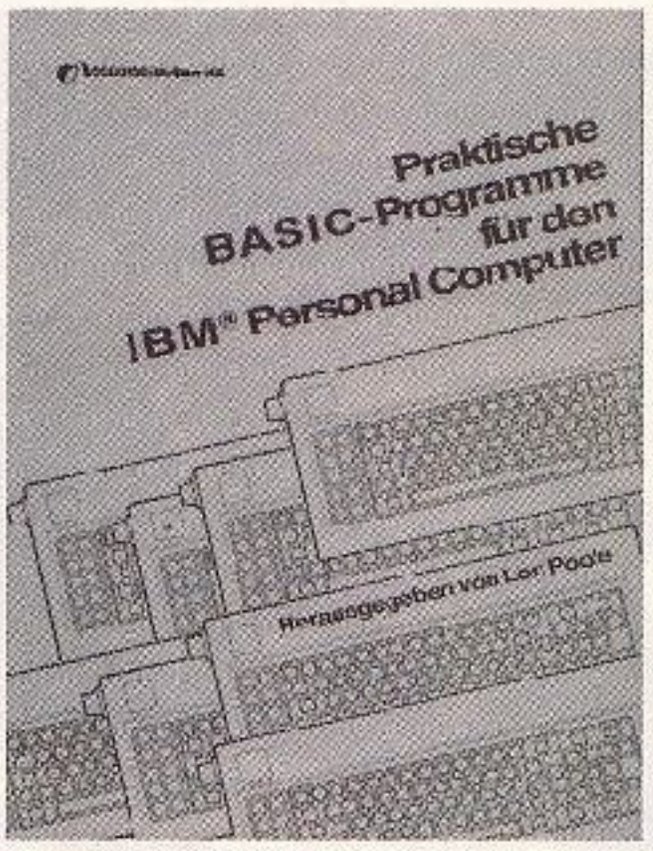
```
10 REM     ********************
20 REM     *    IRRGARTEN     *
30 REM     *                  *
40 REM     ********************
50 REM     ********************
60 REM     *                  *
70 REM     *  COPYRIGHT BY    *
80 REM     *  H.SCHWEITZER    *
90 REM     *                  *
100 REM     *    TI-BASIC     *
110 REM     *                 *
120 REM     ********************
130 CALL CLEAR
140 PRINT "      I R R G A R T E N"::::::::"  fuer Katz u. Maus"::::::::
150 FOR J=1 TO 700
160 NEXT J
170 CALL CLEAR
180 CALL SCREEN(16)
190 FOR J=1 TO 12
200 CALL COLOR(J,2,16)
210 NEXT J
220 PRINT "Der Sinn des Spieles liegt":"darin,die Maus von einem":"Ausgang zum a
nderen"
230 PRINT "Ausgang zu fuehren.":::::"Du hast 5 PUNKTE als BONUS.":::"Ein anstosse
n an die ";
240 PRINT "Einfassung kostet EINEN":"Punkt.Bei Verlassen der":"aeusseren Einfried
ung ist das":::" SPIEL zu ENDE."
250 PRINT :::"WEITER MIT >ENTER<"
260 GOSUB 2160
270 CALL CLEAR
280 PRINT "Aber VORSICHT !":::"den KATZEN musst du aus-":"weichen,":::
290 PRINT "SONST WIRST DU GEFRESSEN!":::"und das Spiel ist zu ENDE.":::
300 PRINT "Blockieren die Katzen alle":"Ausgaenge,so musst du durch":"die Einfas
sung,was einen":"-PUNKT- kostet"
310 PRINT :::"WEITER MIT >ENTER<"
320 GOSUB 2160
330 CALL CLEAR
340 PRINT "GESTREUERT WIRD MIT DEN"::"TASTEN:":::
350 PRINT TAB(5);"Z = links"::TAB(5);"X = rechts"::TAB(5);"> = unten"::TAB(5);"<
 = oben"::::::::
360 PRINT "WEITER MIT >ENTER<"::
370 GOSUB 2160
380 CALL CLEAR
390 CALL SCREEN(14)
400 FOR J=1 TO 9
410 CALL COLOR(J,2,14)
420 NEXT J
430 P=5
440 PRINT TAB(3);"PUNKTE :";P::::::::::::::::::::::::
450 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
460 CALL CHAR(146,"000000FFFF000000")
470 CALL CHAR(147,"0000001818000000")
480 CALL COLOR(15,16,14)
490 CALL HCHAR(3,16,146,2)
500 CALL HCHAR(3,4,128,12)
510 CALL HCHAR(3,18,128,12)
520 CALL HCHAR(24,4,128,21)
530 CALL HCHAR(24,27,128,3)
540 CALL VCHAR(4,4,128,20)
550 CALL VCHAR(3,30,128,22)
560 CALL CHAR(136,"00FFFFFFFFFFFF00")
570 CALL CHAR(137,"3C3C3C3C3C3C3C3C")
580 CALL CHAR(144,"00183C7E7E3C1800")
590 RANDOMIZE
600 FOR I=1 TO 10
610 FOR J=2 TO 7
620 A=INT(RND*29)+1
630 IF A=1 THEN 620 ELSE 640
640 IF A=2 THEN 620 ELSE 650
650 IF A=3 THEN 620 ELSE 660
660 IF A=4 THEN 620
670 CALL COLOR(14,4,14)
680 CALL HCHAR((2*J)-J,A,136,1)
690 NEXT J
700 FOR J=2 TO 7
710 B=INT(RND*29)+1
```

```
720 IF B=1 THEN 710 ELSE 730
730 IF B=2 THEN 710 ELSE 740
740 IF B=3 THEN 710 ELSE 750
750 IF B=4 THEN 710
760 CALL HCHAR((J*2)+J,B,136,1)
770 CALL SOUND(100,(110*J*2),2)
780 NEXT J
790 FOR J=2 TO 9
800 AV=INT(RND*23)+1
810 IF AV=1 THEN 800 ELSE 820
820 IF AV=2 THEN 800 ELSE 830
830 IF AV=3 THEN 800
840 CALL VCHAR(AV,(J*2)+J,137,1)
850 NEXT J
860 FOR J=2 TO 8
870 BV=INT(RND*23)+1
880 IF BV=1 THEN 870 ELSE 890
890 IF BV=2 THEN 870 ELSE 900
900 IF BV=3 THEN 870
910 CALL VCHAR(BV,(J*2)+J,137,1)
920 NEXT J
930 FOR J=1 TO 3
940 CALL SOUND(100,(1047-J*200),0)
950 NEXT J
960 NEXT I
970 GOTO 1200
980 CALL KEY(0,T,Z)
990 IF Z=0 THEN 980
1000 IF T=90 THEN 1040
1010 IF T=88 THEN 1060
1020 IF T=45 THEN 1120
1030 IF T=59 THEN 1160 ELSE 980
1040 GOSUB 2200
1050 H=H-1
1060 CALL SOUND(100,220,0)
1070 GOTO 1230
1080 GOSUB 2200
1090 H=H+1
1100 CALL SOUND(100,110,0)
1110 GOTO 1230
1120 GOSUB 2200
1130 V=V+1
1140 CALL SOUND(100,220,0)
1150 GOTO 1230
1160 GOSUB 2200
1170 V=V-1
1180 CALL SOUND(100,110,0)
1190 GOTO 1230
1200 H=26
1210 V=23
1220 CALL COLOR(15,16,14)
1230 GOSUB 1760
1240 CALL HCHAR(V,H,144)
1250 CALL COLOR(15,16,14)
1260 GOSUB 1550
1270 IF P<1 THEN 1960
1280 KA=INT(RND*4)+1
1290 IF KA=3 THEN 1390
1300 GOTO 980
1310 P=P-1
1320 FOR J=1 TO 5
1330 CALL SOUND(200+J,110,0)
1340 NEXT J
1350 M$=STR$(P)&"          "
1360 FOR J=1 TO 10
1370 CALL HCHAR(1,13+J,ASC(SEG$(M$,J,1)))
1380 NEXT J
1390 CALL CHAR(152,"4244E2713E7F5E44")
1400 CALL COLOR(16,2,14)
1410 FOR J=1 TO 3
1420 ZEILE=INT(RND*23)+1
1430 IF ZEILE=1 THEN 1420 ELSE 1440
1440 IF ZEILE=2 THEN 1420 ELSE 1450
1450 IF ZEILE=3 THEN 1420
1460 SPALTE=INT(RND*28)+1
1470 IF SPALTE=1 THEN 1460 ELSE 1480
```

```
1480 IF SPALTE=2 THEN 1450 ELSE 1490
1490 IF SPALTE=3 THEN 1460 ELSE 1500
1500 IF SPALTE=4 THEN 1460
1510 CALL HCHAR(ZEILE,SPALTE,152)
1520 CALL SOUND(200,880+J*150,0)
1530 NEXT J
1540 GOTO 1300
1550 IF H<5 THEN 1600 ELSE 1560
1560 IF H>29 THEN 1600 ELSE 1570
1570 IF V<4 THEN 1600 ELSE 1580
1580 IF V>23 THEN 1600
1590 RETURN
1600 FOR I=1 TO 15
1610 CALL SOUND(100,1160-I*70,30-I*2)
1620 NEXT I
1630 CALL CLEAR
1640 CALL SCREEN(5)
1650 FOR J=1 TO 9
1660 CALL COLOR(J,16,5)
1670 NEXT J
1680 PRINT "DU HAST DEN IRRGARTEN VER-" :"LASSEN.DAS SPIEL IST AUS."::::::
1690 PRINT "DEINE PUNKTZAHL";P:::::
1700 PRINT :::"EIN NEUES SPIEL ?"::::
1710 PRINT "                J/N"
1720 CALL KEY(0,T,Z)
1730 IF Z=0 THEN 1720
1740 IF T=74 THEN 380 ELSE 1750
1750 IF T=78 THEN 2060 ELSE 1720
1760 CALL GCHAR(V,H,CODE)
1770 IF CODE=136 THEN 1310
1780 IF CODE=137 THEN 1310
1790 IF CODE=152 THEN 1820
1800 IF CODE=146 THEN 2070
1810 RETURN
1820 FOR J=1 TO 10
1830 CALL SOUND(50+J,110,0)
1840 CALL CLEAR
1850 CALL SCREEN(2+J)
1860 NEXT J
1870 CALL CLEAR
1880 CALL SCREEN(7)
1890 FOR J=1 TO 9
1900 CALL COLOR(J,16,7)
1910 NEXT J
1920 PRINT TAB(3);"DU WURDEST";TAB(3);"__________";:TAB(3);"SOEBEN GEFRESSEN!";
1930 PRINT TAB(3);"__________________":::::
1940 PRINT TAB(3);"NOCH EIN VERSUCH ?":::::TAB(13);"J/N":::
1950 GOTO 1720
1960 CALL CLEAR
1970 CALL SCREEN(5)
1980 FOR J=1 TO 9
1990 CALL SOUND(100+J,110+J,0)
2000 CALL COLOR(J,12,5)
2010 NEXT J
2020 PRINT "DU MUSST BESSER"::"AUFPASSEN.":::"DEINE PUNKTZAHL IST"::'ERSCHOEPFT !
":::::::
2030 PRINT "PUNKTZAHL":::"BETRAEGT ";P:::::
2040 PRINT "NOCMAL ?       J/N"::
2050 GOTO 1720
2060 END
2070 CALL CLEAR
2080 CALL SCREEN(6)
2090 FOR J=1 TO 12
2100 CALL COLOR(J,16,6)
2110 NEXT J
2120 PRINT TAB(5);"G R A T U L I E R E";TAB(5);"____________________":::::
2130 PRINT "Du hast es geschafft":::"PUNKTZAHL =";P:::::::
2140 PRINT "Noch ein Spiel ?  J/N"::
2150 GOTO 1720
2160 CALL KEY(0,T,Z)
2170 IF Z=0 THEN 2160
2180 IF T=13 THEN 2190 ELSE 2160
2190 RETURN
2200 CALL COLOR(15,16,14)
2210 CALL HCHAR(V,H,147)
2220 RETURN
```

Bücher

Zu jedem Programm findet der Leser Aufgabestellungen, mehrere Beispiele und einen Probelauf. Für zu Hause, für Übungskurse und für das Büro stehen nützliche Anwendungen preiswert zur Verfügung.
Von der programmiertechnischen Seite vermittelt dieses Buch nicht nur dem Neuling interessante Einblicke, die keineswegs IBM-PC-spezifisch, sondern auch für Anwender kleiner Homecomputer durchaus interessant sind.
Die Autoren gehen deshalb auch ausführlich darauf ein, wie Änderungen durchzuführen sind, erklären Algorythmen und versuchen durch lebens- und realitätsnahe Beispiele in die Materie einzuführen.
Neben den 34 praktischen Programmen sind zusätzlich kleine Tips, die speziell den IBM-Personalcomputer betreffen, vorhanden.



Software Reviews

Pesteroids

von PS-Software

für den TI 99/4A

Das Raumschiff ist vom Kurs abgekommen und befindet sich im Gebiet einer fremden Macht. Welle auf Welle, von feindlichen Abfangjägern, versucht den Eindringling zu vertreiben. Dazu kommen Asteroiden, denen es ausweichen muß.

Die Beschreibung hört sich bekannt an und entspricht im wesentlichen dem üblichen Strickmuster.
Die farbige Grafik ist hervorragend. Man hat hier guten Gebrauch von den Möglichkeiten des TI Extended Basic gemacht. Das gleiche gilt für die Verwendung der Sprites.
Der Sound könnte, zieht man die diesbezüglichen Möglichkeiten des TI 99 in Betracht, sicherlich verbessert werden.
Geübte Spieler werden dieses Programm zudem als sehr einfach empfinden.

99 Vaders

von Stainless Software

für den TI 99/4A

Ein weiteres Programm für den Texas-Computer, ist ein Invader-Spiel. Wer die Invader-Spiele mag, dem wird diese Version ganz bestimmt gefallen. Die Spielidee ist bekannt und braucht nicht weiter erläutert zu werden.

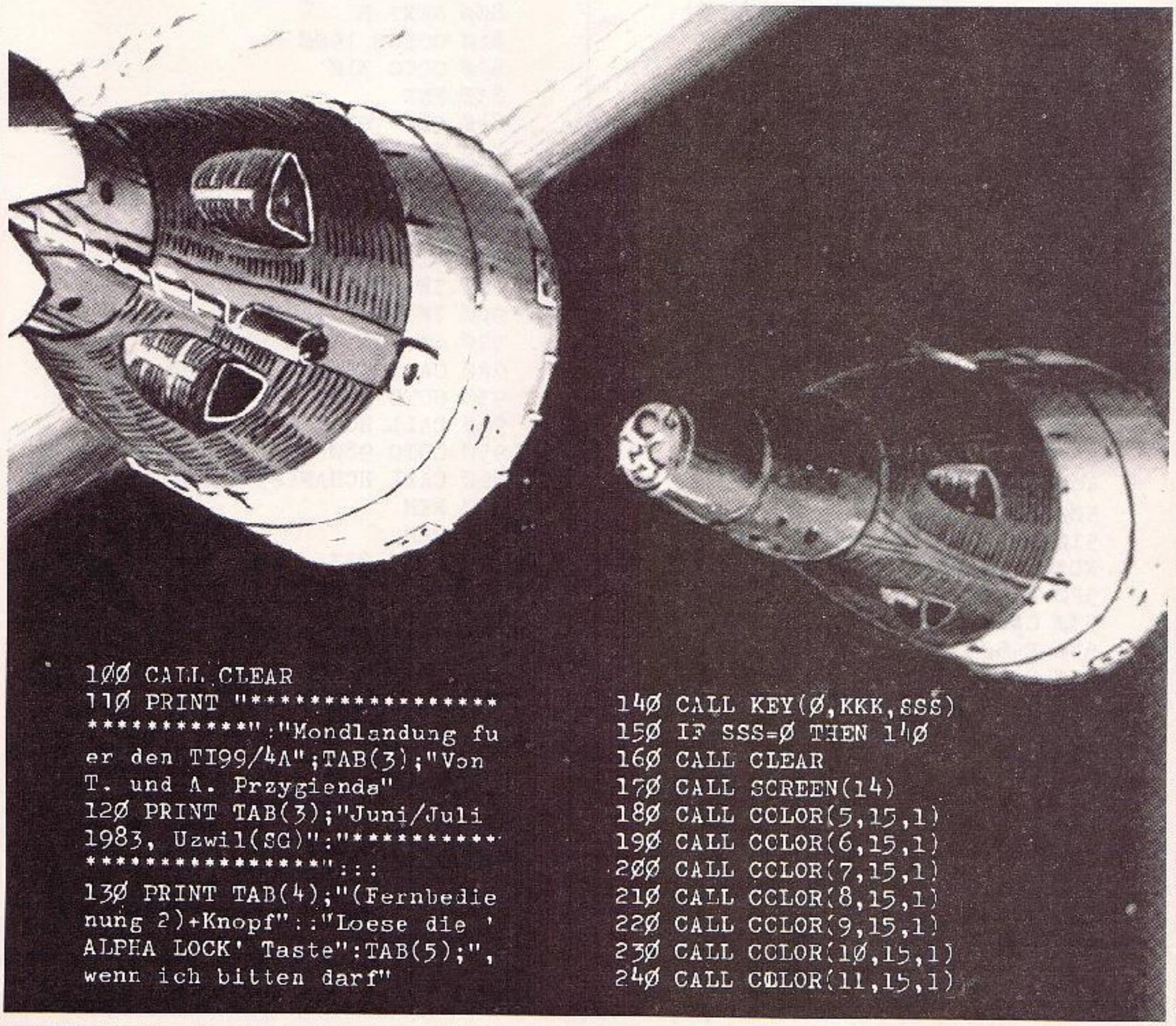
Auch hier sind die grafischen Möglichkeiten des TI 99 gut genutzt worden. Hinzu kommt, eine bei diesem Computer nicht ganz selbstverständliche Spielgeschwindigkeit, zumal hier nur das gewöhnliche TI Basic verwandt wird. Ein guter Sound macht das Programm auch akustisch interessant
Sollte jemand einen Sore von 5000 Punkten erreichen, ist dies wahrlich eine Meisterleistung. Wir waren bereits mit 1000 Punkten zufrieden.

Mondlandung

für den TI-99

Ziel des Spieles ist es, auf einer von den vorhandenen Plattformen zu landen, die in der Landschaft zerstreut liegen. Je dunkler die Plattform, desto tiefer liegt sie zwischen den Bergen und umso mehr Punkte gibt es.

Der Spieler besitzt eine Fähre die mit 500 Einheiten Energie und einer Kananone ausgestattet ist. Sie wird mit Joystick 2 gesteuert. Schiebt man den Hebel nach oben, so bedeutet das: steigen; nach unten: sinken; nach hinten: bremsen und nach vorne: beschleunigen.
Man kann auch gleichzeitig steigen und bremsen - Joystick oben-hinten usw. Jede Betätigung des Hebels benötigt 4 Einheiten Energie; gleichzeitiges Steigen und Bremsen verbraucht 8 Einheiten, und der Schuß 100 Einheiten. Einzukalkulieren ist auch die Trägheit des Schiffes.
Die Landschaft besteht aus Felsen, bei deren Berührung das Schiff explodiert, Wolken, die das Gleiche verursachen, und Meteoriten. Die Landeplätze haben drei Farben. Je dunkler der Landeplatz, desto mehr Punkte bringt er. Ausserdem hängt die Punktevergabe von der Energiemenge, die noch vorhanden ist, ab.
Bei geglückter Landung wird ganz unten in der linken Ecke die Energiemenge angezeigt, in der Mitte die jetzige Punktzahl. Das Spiel geht danach weiter.
Das Schiff driftet auf die Seite und nach unten ab. Bei der Kanone ist zu beachten, daß sie direkt vor dem Schiff nicht wirkt (sie wird erst in einem gewissen Abstand wirksam).



```
100 CALL CLEAR
110 PRINT "********************
**********":"Mondlandung fu
er den TI99/4A";TAB(3);"Von
T. und A. Przygienda"
120 PRINT TAB(3);"Juni/Juli
1983, Uzwil(SG)":"**********
********************":::
130 PRINT TAB(4);"(Fernbedie
nung 2)+Knopf"::"Loese die '
ALPHA LOCK' Taste":TAB(5);",
wenn ich bitten darf"
```

```
140 CALL KEY(0,KKK,SSS)
150 IF SSS=0 THEN 140
160 CALL CLEAR
170 CALL SCREEN(14)
180 CALL CCLOR(5,15,1)
190 CALL CCLOR(6,15,1)
200 CALL CCLOR(7,15,1)
210 CALL CCLOR(8,15,1)
220 CALL CCLOR(9,15,1)
230 CALL CCLOR(10,15,1)
240 CALL CCLOR(11,15,1)
```

```
250 CALL COLOR(12,15,1)
260 CALL CHAR(129,"0451045204
42A4D17")
270 CALL CHAR(130,"241242288
152B0E9")
280 CALL CHAR(131,"2F1F4711A
4104510")
290 CALL CHAR(132,"EAF0E5684
5104411")
300 GOSUB 1810
310 RANDOMIZE
320 X=0
330 Y=0
340 CALL CHAR(145,"0000000AA"
)
350 CALL CHAR(152,"0000000007
E")
360 CALL CHAR(144,"0000000007
E")
370 CALL CHAR(136,"0000000007
E")
380 CALL COLOR(14,16,1)
390 CALL COLOR(15,4,1)
400 CALL CHAR(98,"1840393018
4711041")
410 CALL COLOR(16,2,1)
420 CALL CHAR(120,"FED86FED4
4FE883F")
430 CALL COLOR(12,1,12)
440 CALL COLOR(13,8,1)
450 CALL CHAR(128,"0024 3CFF7
C7C5200")
460 CALL CHAR(133,"1010103 8B
ABAFE82")
470 CALL COLOR(3,16,2)
480 CALL COLOR(4,16,2)
490 CALL HCHAR(1,1,32,746)
500 FOR R99=1 TO 23
510 CALL VCHAR((RND*14)+1,(R
ND*31)+1,128)
520 NEXT R99
530 CALL VCHAR(1,2,32,24)
540 E=500
550 GOSUB 840
560 CALL COLOR(2,12,1)
570 CALL CHAR(96,"F06078FF78
50F")
580 CALL COLOR(9,15,1)
590 A=8
600 B=1
610 R=A
620 V=B
630 B=B+1+X/6
640 A=A-Y/8+.09
650 CALL JOYST(2,X,Y)
660 E=E-ABS(X)-ABS(Y)
670 IF B>32 THEN 690
680 GOTO 700
690 B=E-31
700 CALL HCHAR(R,V,32)
710 CALL GCHAR(A,B,W1)
720 IF W1<>32 THEN 1180
730 CALL HCHAR(A,B,96)
740 IF E<=0 THEN 770
750 GOSUB 1090
760 GOTO 610
770 PRINT ::::::::::"Schade,
 Wegen energiemangel ":"erre
ichst Du den Mond im":"freie
n Fall.":::::::::::TAB(5);"
$$$$$Bumm$$$%^"
780 PRINT ::"Es entstand ein
 neuer Krater":"mit";INT(RND
*1000);" Meter Tiefe.":
790 FOR B=1 TO 1000
800 NEXT B
810 GOSUB 1900
820 GOTO 310
830 END
840 REM  BERGE
850 FOR B33=1 TO 32
860 V=INT(RND*12)+1
870 CALL VCHAR(24-V,B33,120,
V)
880 V1=INT(RND*10)
890 IF V1<7 THEN 990
900 IF V>10 THEN 960
910 IF V>6 THEN 940
920 CALL HCHAR(23-V,B33,152)
930 GOTO 980
940 CALL HCHAR(23-V,B33,141)
950 GOTO 980
960 CALL HCHAR(23-V,B33,136)
970 REM
980 REM
990 NEXT B33
1000 REM wolken
1010 FOR V=1 TO 4
1020 FOR V2=1 TO 5
1030 V3=INT(RND*31)+1
1040 V4=INT(RND*10)
1050 CALL HCHAR(V2,V3,98,V4)
1060 NEXT V2
1070 NEXT V
1080 RETURN
1090 REM schuss
1100 CALL KEY(2,K1,S)
1110 IF K1<>18 THEN 1170
1120 IF B>30 THEN 1170
1130 E=E-100
1140 CALL HCHAR(A,B-2,145,3)
1150 CALL SOUND(-90,781,0,83
```

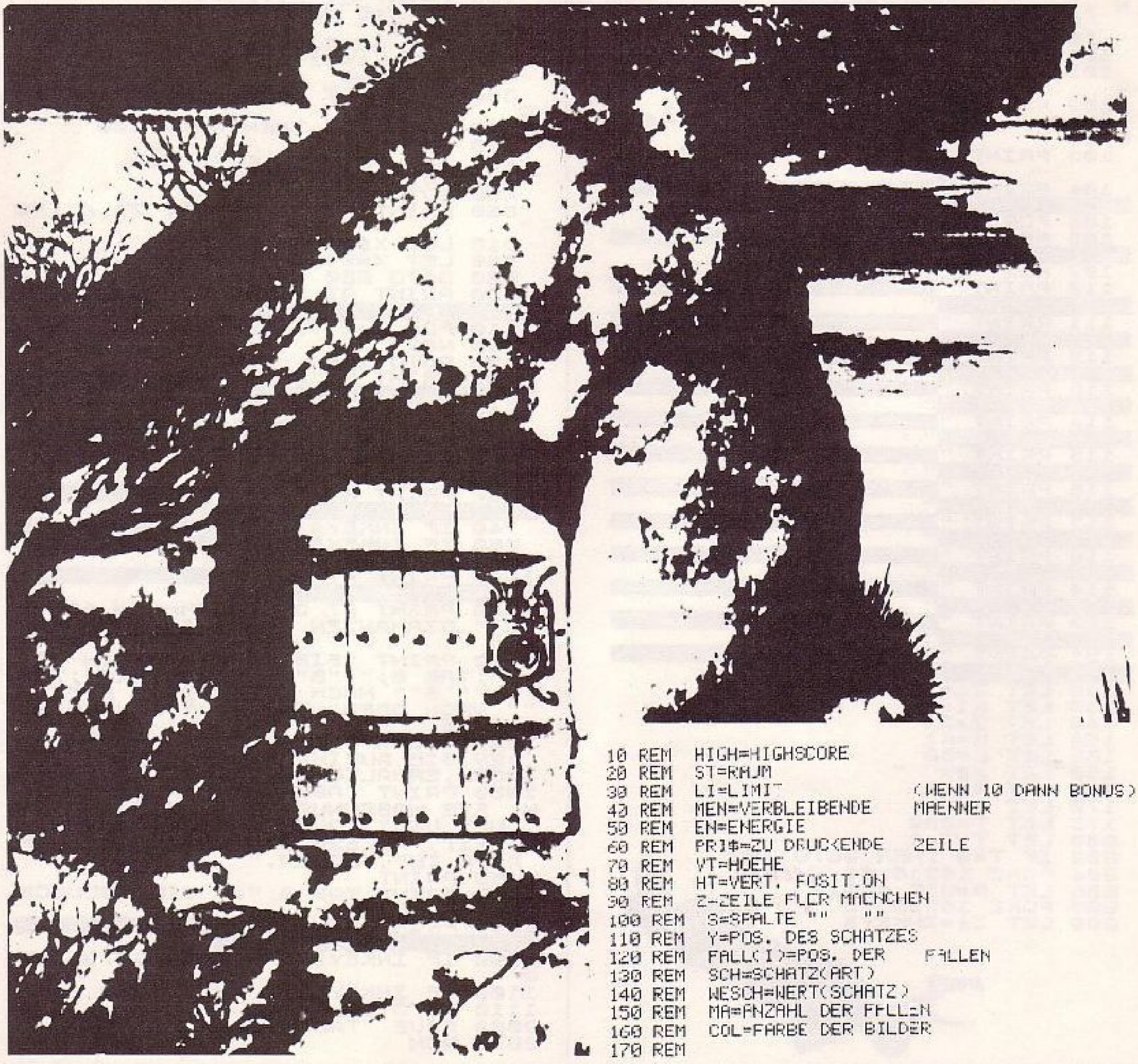
```
Ø,2,852,5,-1,3)
116Ø CALL HCHAR(A,B+2,32,3)
117Ø RETURN
118Ø REM EXPLOSION?
119Ø IF W1<>152 THEN 122Ø
12ØØ GOSUB 131Ø
121Ø GOTO 127Ø
122Ø IF W1<>144 THEN 125Ø
123Ø GOSUB 138Ø
124Ø GOTO 127Ø
125Ø IF W1<>136 THEN 128Ø
126Ø GOSUB 145Ø
127Ø GOTO 49Ø
128Ø GOSUB 152Ø
129Ø CALL CLEAR
13ØØ GOTO 31Ø
131Ø REM landung NA I
132Ø CALL HCHAR(A,B,133)
133Ø P=P+E*3Ø
134Ø CALL SOUND(5Ø,44Ø,Ø,-1,
5,11Ø,3)
135Ø CALL SOUND(8Ø,333,Ø,-3,
5,22Ø,3)
136Ø GOSUB 163Ø
137Ø RETURN
138Ø REM LANDUNG NA II
139Ø CALL HCHAR(A,B,133)
14ØØ CALL SOUND(5Ø,33Ø,Ø,-1,
5,22Ø,3)
141Ø CALL SOUND(8Ø,222,Ø,-2,
5,33Ø,3)
142Ø P=P+E*2Ø
143Ø GOSUB 163Ø
144Ø RETURN
145Ø REM LANDUNG NA III
146Ø CALL HCHAR(A,B,133)
147Ø CALL SOUND(5Ø,44Ø,Ø,-3,
5,33Ø,3)
148Ø CALL SOUND(8Ø,333,Ø,-3,
5,44Ø,3)
149Ø P=P+E*1Ø
15ØØ GOSUB 163Ø
151Ø RETURN
152Ø REM EXPLOSION
153Ø CALL COLOR(13,15,1)
154Ø CALL HCHAR(A,B,131)
155Ø CALL HCHAR(A-1,B,129)
156Ø CALL HCHAR(A,B+1,132)
157Ø CALL HCHAR(A-1,B+1,13Ø)
158Ø CALL SOUND(15Ø,-8,Ø,44Ø
,15)
159Ø CALL SOUND(3ØØ,-7,Ø,33Ø
,1Ø)
16ØØ CALL SOUND(6ØØ,-6,Ø,22Ø
,1Ø)
161Ø GOSUB 19ØØ
162Ø RETURN
163Ø REM ZAEHLER
164Ø REM ENERGIE
165Ø E$=STR$(E)
166Ø FOR V1=1 TO LEN(E$)
167Ø IF E$="" THEN 171Ø
168Ø EA$=SEG$(E$,V1,1)
169Ø CALL HCHAR(24,V1+2,ASC(
EA$))
17ØØ NEXT V1
171Ø REM PUNKTE
172Ø P$=STR$(P)
173Ø FOR V1=1 TO LEN(P$)
174Ø IF P$="" THEN 178Ø
175Ø PA$=SEG$(P$,V1,1)
176Ø CALL HCHAR(24,V1+15,ASC
(PA$))
177Ø NEXT V1
178Ø FOR G=1 TO 1ØØØ
179Ø NEXT G
18ØØ RETURN
181Ø REM GRAPHIK
182Ø RESTORE 183Ø
183Ø KK$="     Also fangen w
ir an    "
184Ø G=1
185Ø FOR B=3 TO LEN(KK$)
186Ø KR$=SEG$(KK$,B,1)
187Ø CALL HCHAR(G,B,ASC(KR$)
)
188Ø NEXT B
189Ø RETURN
19ØØ REM            **Fertig**
191Ø REM
192Ø CALL CLEAR
193Ø IF P<HIGSC THEN 2ØØØ
194Ø HIGSC=P
195Ø CALL SOUND(6Ø,55Ø,Ø,44Ø
,Ø,22Ø,8)
196Ø CALL SOUND(1ØØ,44Ø,Ø,55
Ø,Ø,33Ø,8)
197Ø CALL SOUND(8Ø,33Ø,Ø,44Ø
,Ø,315,8)
198Ø REM
199Ø REM
2ØØØ PRINT "Purkte ";TAB(2Ø)
;P::::::"Highscore ";TAB(2Ø)
;HIGSC::::::
2Ø1Ø PRINT "Noch ein Spiel (
j/n)  "
2Ø2Ø CALL KEY(Ø,K,S)
2Ø3Ø IF S=Ø THEN 2Ø2Ø
2Ø4Ø P=Ø
2Ø5Ø IF K=1Ø6 THEN 16Ø
2Ø6Ø STOP
2Ø7Ø RETURN
```

Schatzsuche

für den TI-99

Besitzer von Telespielen sind an actiongeladene Spiele gewöhnt. Homecomputeranwender, die selbst programmieren, müssen schon eine Menge über den Gebrauch von Maschinenspracheroutinen wissen, um Gleichwertiges zu erreichen. Manchmal gelingt einem jedoch auch ohne in Assembler programmieren zu müssen, ein actionreiches, schnelles Spielprogramm in Basic zu entwickeln.
Schatzsuche für den TI-99 kann man sicherlich als ein gut gelungenes Abenteuerspiel mit interessanten Szenen und viel Action bezeichnen.

Jonathan, der Held des Spieles – der Spieler steuert ihn über die Tasten:
D = laufen
Space = (einen Zweischrittesprung machen) und
E = drei Schritte aufeinmal springen – mu3 auf seinem Weg zum Ziel, den Schatz zu finden, viele Gefahren überstehen, wofür dem Spieler Punkte gutgeschrieben werden.
Laufen und springen verbraucht natürlich Energie, mit der man gut umgehen muß.
Spinnen, Krokodile und Fallen tun ein übriges, daß dieses Spiel nicht langweilig wird.
Schatzsuche läuft auf der Grundversion TI-99/4A ohne ext. Basic.
Zum Schluß noch ein kleiner Tip für den eiligen Anwender, der als weniger geübter Spieler auch mal gerne das Ziel erreichen
möchte: Ändern Sie die Zeile 1950!



```
10 REM  HIGH=HIGHSCORE
20 REM  ST=RAUM
30 REM  LI=LIMIT                    (WENN 10 DANN BONUS)
40 REM  MEN=VERBLEIBENDE    MAENNER
50 REM  EN=ENERGIE
60 REM  PRI$=ZU DRUCKENDE   ZEILE
70 REM  VT=HOEHE
80 REM  HT=VERT. POSITION
90 REM  Z=ZEILE FUER MAENNCHEN
100 REM  S=SPALTE "     "
110 REM  Y=POS. DES SCHATZES
120 REM  FALL(I)=POS. DER    FALLEN
130 REM  SCH=SCHATZ(ART)
140 REM  WESCH=WERT(SCHATZ)
150 REM  MA=ANZAHL DER FALLEN
160 REM  COL=FARBE DER BILDER
170 REM
```

```
200 REM  ZEICHENDEFINITION
210 GOSUB 1660
220 CALL CLEAR
230 CALL SCREEN(14)
240 HIGH=9999
250 ST=1
260 FOR I=1 TO 16
270 CALL COLOR(I,1,1)
280 NEXT I
290 REM
300 PRINT "WILLKOMMEN ZUR SCHATZSUCHE"::::"BRAUCHST DU ANLEITUNGEN?"::"(J) (N) D
RUECKEN"::::
310 REM  ZEICHEN ERSCHEINEN
320 FOR I=1 TO 16
330 CALL COLOR(I,16,1)
340 NEXT I
350 CALL KEY(0,KEY,STATUS)
360 CALL SOUND(-300,330,0,440,0,550,0)
370 CALL SOUND(-100,660,0,880,0,1111,0)
380 CALL SOUND(-300,330,0,800,0,550,0)
390 IF STATUS=0 THEN 350
400 IF KEY=74 THEN 430
410 IF KEY=78 THEN 530
420 GO TO 350
430 CALL CLEAR
440 REM
450 FOR I=1 TO 16
460 CALL COLOR(I,1,1)
470 NEXT I
480 PRINT "VIELE PUNKTE HOLEN BEVOR DIE"::"ENERGIE ALLE IST! ABER ES"::
490 PRINT "SIND GEFAHREN AUF DEM WEG"::"PUNKTE BRINGEN:  b c d e f 9"::
500 PRINT "DIE ANZAHL DER PUNKTE"::"IST ABHAENGIG VON DER FARBE"::
510 PRINT "GESTEUERT WIRD MIT:"::"SPACE=SPRUNG (2 SCHRITTE)"::"E    =SPRUNG (3 S
CHRITTE)"::
520 PRINT "D    =LAUFEN (1 SCHRITT)"::
530 PRINT "PRESS ANY KEY TO START"::
540 REM  ZEICHEN                              ERSCHEINEN                    WIEDER
550 FOR I=1 TO 16
560 CALL COLOR(I,16,1)
570 NEXT I
580 CALL KEY(0,KEY,STATUS)
590 CALL SOUND(-100,110,0,330,0,660,0)
600 CALL SOUND(-100,330,0,990,0,1200,0)
610 IF STATUS=0 THEN 580
620 IF KEY<>1 THEN 630
630 REM
640 LI=10
650 MEN=5
660 EN=2000
670 CALL CHAR(32,"")
680 CALL CLEAR
690 REM  SCHLEIFE LAESST        ZEICHEN ERST ERSCHEINEN     WENN ALLE AUF DEM
         BILDSCHIRM SIND
700 FOR I=3 TO 16
710 CALL COLOR(I,1,1)
720 NEXT I
730 REM
740 REM  ANZEIGE
750 PRI$='POINTS"
760 VT=2
770 HT=3
780 GOSUB 2060
790 PRI$="ENERGIE"
800 HT=19
810 GOSUB 2060
820 HT=3
830 VT=4
840 PRI$='HIGH"
850 GOSUB 2060
860 GOSUB 2160
870 HT=15
880 PRI$='ROOM"
890 GOSUB 2060
900 GOSUB 2200
910 GOSUB 2100
920 GOSUB 2240
```

```
930 HT=23
940 PRI$='MEN"
950 GOSUB 2060
960 REM  ZEIGT NOCH                    VERBLEIBENDE MAENNER AN
970 CALL HCHAR(4,27,112,MEN-1)
980 Z=15
990 S=19
1000 CALL HCHAR(16,1,121,32)
1010 CALL HCHAR(1,1,121,32)
1020 CALL HCHAR(5,1,121,32)
1030 CALL HCHAR(24,1,121,32)
1040 CALL HCHAR(3,1,121,32)
1050 GOSUB 4340
1060 CALL HCHAR(Z,S,112)
1070 CALL KEY(5,KEY,STATUS)
1080 CALL SOUND(-100,110,4,-3,2)
1090 IF STATUS=0 THEN 1070
1100 IF KEY=69 THEN 1440
1110 IF KEY=68 THEN 1150
1120 IF KEY=32 THEN 1280
1130 GO TO 1070
1140 REM   LAUF+WERTUNG
1150 REM
1160 CALL HCHAR(Z,S,113)
1170 CALL HCHAR(Z,S,114)
1180 CALL HCHAR(Z,S,32)
1190 S=S+1
1200 IF S>31 THEN 3010
1210 CALL HCHAR(Z,S,115)
1220 EN=EN-5
1230 IF S=Y THEN 3350 ELSE 1240
1240 FOR I=1 TO MA
1250 IF S=FALL(I)THEN 2460
1260 NEXT I
1270 GO TO 1060
1280 CALL HCHAR(Z,S,113)
1290 CALL HCHAR(Z,S,32)
1300 S=S+1
1310 IF S>31 THEN 3010
1320 CALL HCHAR(Z-1,S,114)
1330 CALL HCHAR(Z-1,S,32)
1340 S=S+1
1350 IF S>31 THEN 3010
1360 CALL HCHAR(Z-1,S,115)
1370 CALL HCHAR(Z-1,S,32)
1380 EN=EN-20
1381 CALL HCHAR(Z,S,115)
1390 IF S=Y THEN 3350 ELSE 1400
1400 FOR I=1 TO MA
1410 IF S=FALL(I)THEN 2460
1420 NEXT I
1430 GO TO 1060
1440 CALL HCHAR(Z,S,113)
1450 CALL HCHAR(Z,S,32)
1460 S=S+1
1470 IF S>31 THEN 3010
1480 CALL HCHAR(Z-2,S,113)
1490 CALL HCHAR(Z-2,S,32)
1500 S=S+1
1510 IF S>31 THEN 3010
1520 CALL HCHAR(Z-1,S,114)
1530 CALL HCHAR(Z-1,S,32)
1540 S=S+1
1550 IF S>31 THEN 3010
1560 CALL HCHAR(Z-1,S,115)
1570 CALL HCHAR(Z-1,S,32)
1580 EN=EN-35
1590 CALL HCHAR(Z-1,S,114)
1600 CALL HCHAR(Z-1,S,32)
1601 CALL HCHAR(Z,S,115)
1610 IF S=Y THEN 3350 ELSE 1620
1620 FOR I=1 TO MA
1630 IF S=FALL(I)THEN 2460
1640 NEXT I
1650 GO TO 1060
1660 REM  SPINNE
1670 CALL CHAR(97,"0024247E427E4281")
```

```
1680 REM   URNE
1690 CALL CHAR(98,"000018C37E7E3C18")
1700 REM  FALLE
1710 CALL CHAR(120,"AFAAAAAAAAAAAFAFF")
1720 REM   BARREN
1730 CALL CHAR(100,"0000FE7E7E7E7E0000")
1740 REM  RING
1750 CALL CHAR(101,"3C18244242241800")
1751 REM  DIAMANT
1752 CALL CHAR(99,"183C7EFFFF7E3C18")
1753 REM  ZEPTER
1754 CALL CHAR(103,"1818183C183C7E7E3C")
1755 REM  BECHER
1756 CALL CHAR(102,"003C3E3D3D3E3C7E")
1760 REM  KROKODIL (ZU)
1770 CALL CHAR(131,"00000004FBAF57FF")
1780 REM  KROKODIL (OFFEN)
1790 CALL CHAR(132,"00C0A0341B0757FF")
1800 REM  TARZAN (STAND)
1810 CALL CHAR(112,"1819FFBC3C3C2424")
1820 REM  TARZAN (2)
1830 CALL CHAR(113,"1891FF3D3C3CE408")
1840 REM  TARZAN (3)
1850 CALL CHAR(114,"1891FF3D3C3CE402")
1860 REM  TARZAN (4)
1870 CALL CHAR(115,"1891FF3D3C3CE404")
1880 REM  TRENNBALKEN
1890 CALL CHAR(121,"FFFFAAAAAAAAFFFF")
1900 RETURN
1910 FOR I=1 TO 4
1920 CALL SOUND(-100,230,0,440,0,550,0)
1930 CALL SOUND(-200,500,0,600,0,700,0)
1940 NEXT I
1950 IF LI>39 THEN 3650
1960 IF LI=20 THEN 1970 ELSE 2000
1970 MEN=MEN+1
1980 CALL SOUND(-500,560,0,780,0,999,0)
1990 CALL HCHAR(4,27,112,MEN-1)
2000 POI=POI+150*LI
2010 EN=EN+750
2020 GOSUB 2100
2030 LI=LI+10
2040 GOSUB 2240
2050 GO TO 3080
2060 FOR UI=1 TO LEN(PRI$)
2070 CALL HCHAR(VT,HT+UI-1,ASC(SEG$(PRI$,LI,1)))
2080 NEXT UI
2090 RETURN
2100 FOR UI=1 TO LEN(STR$(POI))
2110 CALL HCHAR(2,9+UI,ASC(SEG$(STR$(POI),UI,1)))
2120 NEXT UI
2130 IF POI>=HIGH THEN 2150 ELSE 2140
2140 RETURN
2150 HIGH=POI
2160 FOR UI=1 TO LEN(STR$(HIGH))
2170 CALL HCHAR(4,7+UI,ASC(SEG$(STR$(HIGH),UI,1)))
2180 NEXT UI
2190 RETURN
2200 FOR UI=1 TO LEN(STR$(ST))
2210 CALL HCHAR(4,19+UI,ASC(SEG$(STR$(ST),UI,1)))
2220 NEXT UI
2230 RETURN
2240 EN$=STR$(EN)
2250 IF LEN(EN$)>3 THEN 2280
2260 EN$="0"&EN$
2270 GO TO 2250
2280 FOR UI=1 TO 4
2290 CALL HCHAR(2,26+UI,ASC(SEG$(EN$,UI,1)))
2300 NEXT UI
2310 IF EN<5 THEN 2315 ELSE 2320
2315 S=25
2316 GOTO 2460
2320 RETURN
2330 RANDOMIZE
2340 Y=INT((28-4+1)*RND)-4
2350 RANDOMIZE
2360 SCH=INT(6*RND)+1
```

```
2370 IF SCH=1 THEN 2410
2380 IF SCH=2 THEN 2430
2381 IF SCH=3 THEN 2421
2382 IF SCH=4 THEN 2423
2383 IF SCH=5 THEN 2425
2390 WESCH=98
2400 GO TO 2440
2410 WESCH=100
2420 GO TO 2440
2421 WESCH=99
2422 GOTO 2440
2423 WESCH=102
2424 GOTO 2440
2425 WESCH=103
2426 GOTO 2440
2430 WESCH=101
2440 CALL HCHAR(15,Y,WESCH)
2450 GO TO 1210
2460 FOR I=15 TO 23 STEP 1
2470 CALL SOUND(-200,110,0,110,0,110,0,-8,0)
2480 CALL HCHAR(I,S,32)
2490 CALL HCHAR(I,S,112)
2500 CALL HCHAR(I,S,113)
2510 CALL HCHAR(I,S,114)
2520 CALL HCHAR(I,S,32)
2530 CALL HCHAR(I,S,115)
2540 CALL HCHAR(I,S,112)
2550 CALL HCHAR(I,S,32)
2560 NEXT I
2570 CALL HCHAR(23,S,112)
2580 FOR I=32 TO S STEP -1
2590 CALL HCHAR(23,I,131)
2600 CALL HCHAR(23,S,112)
2610 CALL SOUND(-10,-8,0,110,6,190,8)
2620 CALL HCHAR(23,I,132)
2630 CALL SOUND(-10,-1,0,110,6,290,6)
2640 CALL HCHAR(23,I,32)
2650 CALL HCHAR(23,S,115)
2660 NEXT I
2670 CALL HCHAR(23,S,132)
2680 CALL HCHAR(23,S,131)
2690 FOR I=1 TO 4
2700 CALL SOUND(-99,-8,0,110,3,110,3)
2710 CALL SOUND(-111,-4,0,110,8,560,4)
2720 NEXT I
2730 MEN=MEN-1
2740 IF MEN<1 THEN 3400
2750 FOR I=1 TO MA
2760 FALL(I)=0
2770 NEXT I
2780 GO TO 660
2790 REM  FALLEN
2800 FOR I=1 TO MA
2810 RANDOMIZE
2820 FALL(I)=INT((28-3+1)*RND)+3
2830 CALL HCHAR(15,FALL(I),131)
2840 NEXT I
2850 S=2
2860 GO TO 2330
2870 FOR I=1 TO MA
2880 RANDOMIZE
2890 FALL(I)=INT((28-3+1)*RND)+3
2900 CALL HCHAR(16,FALL(I),120)
2910 NEXT I
2920 S=2
2930 GO TO 2330
2940 FOR I=1 TO MA
2950 RANDOMIZE
2960 FALL(I)=INT((28-3+1)*RND)+3
2970 CALL HCHAR(15,FALL(I),97)
2980 NEXT I
2990 S=2
3000 GO TO 2330
3010 RANDOMIZE
3020 GOSUB 2240
3030 CALL HCHAR(15,1,32,32)
```

```
3040 CALL HCHAR(16,1,121,32)
3050 ST=ST+1
3060 GOSUB 2200
3070 IF ST=L1 THEN 1910
3080 REM
3090 RANDOMIZE
3100 COL=INT((14-2+1)*RND)+2
3110 COL2=INT((16-3+1)*RND)+3
3120 IF COL2<>COL THEN 3130 ELSE 3100
3130 COL3=INT((COL2+COL)/2)
3140 IF INT(COL3)<>COL THEN 3150 ELSE 3100
3150 IF INT(COL3)<>COL2 THEN 3160 ELSE 3100
3160 CALL COLOR(9,COL,1)
3170 CALL COLOR(10,COL,1)
3180 CALL COLOR(12,COL2,1)
3190 RANDOMIZE
3200 MA=INT((7-4+1)*RND)+4
3210 RANDOMIZE
3220 PIC=INT(4*RND)+1
3230 CALL SCREEN(COL3)
3240 IF PIC=1 THEN 2790
3250 IF PIC=2 THEN 2870
3260 IF PIC=3 THEN 2940
3270 REM    1.FALLE
3280 FOR I=1 TO MA
3290 RANDOMIZE
3300 FALL(I)=INT((28-3+1)*RND)+3
3310 CALL HCHAR(15,FALL(I),131)
3320 NEXT I
3330 S=2
3340 GO TO 2330
3350 CALL SOUND(-100,330,0,440,0,550,0)
3360 CALL SOUND(-200,440,0,550,0,660,0)
3370 POI=POI+(100*SCH)*COL
3380 GOSUB 2100
3381 IF WESCH=102 THEN 3382 ELSE 3390
3382 CALL SOUND(-300,450,0,656,0,720,0)
3383 EN=EN+350
3384 GOSUB 2240
3390 GO TO 1270
3400 FOR I=1 TO 4
3410 CALL SOUND(500,320,0,430,0,540,0)
3420 CALL SOUND(1000,110,0,240,0,260,0)
3430 CALL SOUND(500,300,0,240,0,110,0)
3440 CALL SOUND(1000,210,0,190,0,300,0)
3450 NEXT I
3460 FOR I=1 TO 100
3470 NEXT I
3480 CALL CLEAR
3490 PRINT "DU HAST ALLE MAENNER":"VERLOREN":::"DU HAST";POI;"GEHOLT":::
3500 PRINT "NOCHMAL - TASTE DRUECKEN":::
3510 IF POI>=HIGH THEN 3520 ELSE 3560
3520 HIGH=POI
3530 FOR I=1 TO MA
3540 FALL(I)=0
3550 NEXT I
3560 CALL KEY(0,KEY,STATUS)
3570 POI=0
3580 ST=1
3590 CALL CHAR(32,"")
3600 IF STATUS=0 THEN 3560
3610 IF KEY<>1 THEN 630
3620 END
3630 END
3640 REM
3650 FOR I=1 TO 3
3660 CALL SOUND(300,450,0,650,0,700,0)
3670 CALL SOUND(100,200,0,110,0,320,0)
3680 CALL SOUND(300,400,0,450,0,900,0)
3690 CALL SOUND(100,110,0,600,0,790,0)
3700 NEXT I
3710 FOR I=1 TO 3
3720 CALL SOUND(300,450,0,1200,0,3200,0)
3730 CALL SOUND(100,1200,0,450,0,900,0)
3740 CALL SOUND(300,3500,0,2100,0,4000,0)
3750 CALL SOUND(100,110,0,900,0,1500,0)
3760 NEXT I
```

```
3770 FOR I=1 TO 50
3780 NEXT I
3790 CALL HCHAR(15,14,121,9)
3800 CALL SOUND(-500,110,0,200,0,250,0)
3810 CALL HCHAR(14,15,121,7)
3820 CALL SOUND(-500,200,0,250,0,300,0)
3830 CALL HCHAR(13,16,121,5)
3840 CALL SOUND(-500,250,0,300,0,400,0)
3850 CALL HCHAR(12,17,121,3)
3860 CALL SOUND(-500,300,0,400,0,550,0)
3870 CALL HCHAR(11,18,121)
3880 CALL SOUND(-500,400,0,550,0,600,0)
3890 FOR I=1 TO 13
3900 CALL HCHAR(15,I,112)
3910 CALL HCHAR(15,I,113)
3920 CALL HCHAR(15,I,114)
3930 CALL HCHAR(15,I,115)
3940 CALL HCHAR(15,I,32)
3950 CALL SOUND(-100,110,0,-4,0)
3960 NEXT I
3970 FOR I=112 TO 115
3980 CALL HCHAR(14,14,I)
3990 NEXT I
4000 CALL HCHAR(14,14,32)
4010 CALL SOUND(-400,330,0,560,0,780,0)
4020 FOR I=112 TO 115
4030 CALL HCHAR(13,15,I)
4040 NEXT I
4050 CALL HCHAR(13,15,32)
4060 CALL SOUND(-500,436,0,780,0,1200,0)
4070 FOR I=112 TO 115
4080 CALL HCHAR(12,16,I)
4090 NEXT I
4100 CALL HCHAR(12,16,32)
4110 CALL SOUND(-500,1200,0,1400,0,1999,0)
4120 FOR I=112 TO 115
4130 CALL HCHAR(11,17,I)
4140 NEXT I
4150 CALL SOUND(-2000,1900,0,2100,0,3500,0)
4160 CALL HCHAR(11,17,32)
4170 FOR FF=1 TO 40
4180 FOR I=112 TO 115
4190 CALL HCHAR(10,18,I)
4200 NEXT I
4210 CALL CHAR(32,"0000000001000000")
4220 CALL SOUND(-500,330,0,550,0,890,0)
4230 CALL CHAR(32,"")
4240 CALL SOUND(-1000,450,0,590,0,1200,0)
4250 CALL SOUND(-500,330,0,999,0,1500,0)
4260 NEXT FF
4270 FOR I=1 TO 200
4280 NEXT I
4290 REM
4300 CALL CLEAR
4310 PRINT "DU HAST ES GESCHAFFT"::"DU BIST AM ZIEL ANGEKOMMEN":::"AUF DEINEM W
EG HAST DU"::;POI
4320 PRINT ::"PUNKTE GESAMMELT":"DAS IST BISHER NUR WENIGEN"::"GELUNGEN"::
4330 GO TO 3500
4340 FOR I=1 TO 4
4350 CALL SOUND(-100,330,0,550,0,780,0)
4360 CALL SOUND(-100,400,0,900,0,1200,0)
4370 CALL SOUND(-100,300,0,670,0,1000,0)
4380 NEXT I
4390 FOR I=3 TO 8
4400 CALL COLOR(I,2,16)
4410 NEXT I
4420 CALL COLOR(10,13,1)
4430 CALL COLOR(12,16,1)
4440 CALL COLOR(11,16,1)
4444 CALL COLOR(13,2,1)
4450 CALL CHAR(32,"0000000001300000")
4460 RETURN
32000 BYTE=BYTE+8
32101 GOSUB 32000
```

Night-Fight

Für den TI-99/4A

Nacht-Kampf - ein Spiel für den TI-99/4A mit einem tollen Bildschirmaufbau und gutem Sound. Die Angreifer fliegen zur nächtlichen Stunde über die Stadt und werden sie rücksichtslos zerstören, wenn Sie es nicht schaffen sie aufzuhalten und abzuschießen.

Dies ist aber nur dann möglich, wenn das feindliche Kampfflugzeug von dem Radarstrahl, der von der Erde ausgeht, erfaßt wird und Sie im richtigen Augenblick mit Ihrem Panzer den vernichtenden Schuß abfeuern.
Da dies sehr schnell geschieht, heißt es wachsam sein und den Joystick - über den das Spiel gesteuert wird - fest im Griff zu haben.
Gelingt es nicht, ein Flugzeug zu treffen, wirft dies bei jedem Flug eine Bombe über der Stadt ab, die somit langsam aber sicher dem Erdboden gleich gemacht wird und das Spiel dann beendet ist.
Auf dem Bildschirm werden nun Ihre Abschüsse angezeigt und der Computer fragt, ob Sie ein neues Spiel möchten.

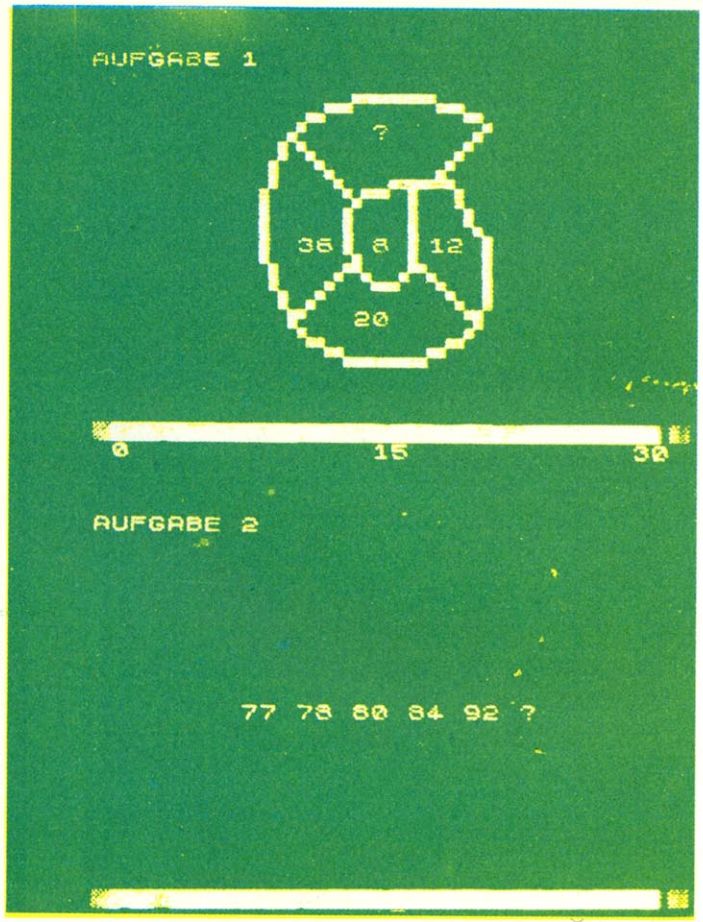
```
100 CALL CLEAR
110 CALL CHAR(136,"3C4299A1A199423C")
120 DISPLAY AT(8,6):"N I G H T   F I G H T" :: DISPLAY AT(13,11):"BY ZIEGLER" ::
 DISPLAY AT(14,9):"ERZGEBIRGSTR.1"
130 DISPLAY AT(15,9):"8052 MOOSBURG" :: CALL HCHAR(13,11,136)
140 FOR I=1 TO 400 :: NEXT I
150 PRINT "SIE KAEMPFEN IM LETZTEN":"SPERRGUERTEL DER DIE STADT":"BESCHUETZT." :
: PRINT
160 PRINT "WENN ES IHNEN NICHT":"GELINGT DIE ANGREIFER":"ABZUSCHIESSEN, WIRD DIE
":"STADT LANGSAM ELIMINIERT."
170 PRINT :: PRINT "MIT DEN JOYSTICKS":"KANN DAS PANZERGESCHUETZ":"AUF DAS ZIEL
EINGERASTET":"WERDEN."
180 PRINT :: PRINT :: PRINT ">>>VIEL GLUECK<<<" :: PRINT :: PRINT :: PRINT "ALPH
A LOCK TASTE LOESEN" :: PRINT :: PRINT 'PRESS ENTER"
190 CALL KEY(0,K,S):: IF S=0 THEN 190
200 CALL CLEAR
210 CALL SCREEN(2)
220 CALL MAGNIFY(3)
230 CALL COLOR(12,4,1,8,16,7)
```

```
240 CALL COLOR(4,13,1,3,7,1)
250 CALL COLOR(9,6,1,6,7,1)
260 CALL CHAR(48,"0F0301010101010103F0C080808080808001010101030F3FFF8080808080C0F0FC
FF")
270 CALL CHAR(56,"C1070107070F01C071F01071F3F01073FFF80E080E0F080E0F880E0F8FC80E0FC
FF")
280 CALL CHAR(64,"C01F1F07C7E1C7FFF7FE1C7071F1F000000000080C0E0F0FEFFFEF0E0C0000000
00")
290 CALL CHAR(68,"0000000E060303030303030101000000000000E0C0808080808080E0C0000000
00")
300 CALL CHAR(72,"FFF77F3C1F0F03010307070F0F5F5FFFFFEFDE3CF8F0C080C0E0E0F0F0FAFAFB
FF")
310 CALL CHAR(80,"FFFFFFFFFFFFFFFF")
320 CALL CHAR(95,"00FF999999FFFFFF")
330 CALL CHAR(96,"4040404040404040403F7FFFC3C3FFFFFFE0E0F0F0F8F8F8F83F7FE0CEDFDBDF
0E")
340 CALL CHAR(100,"FFFFC11C3C063E1CFFFF83387C6C7C38F8FC0673F9D9F971")
350 CALL CHAR(104,"0000000000000000000000FFFF0000003F0000000000000000000000FFFF000
0FF")
360 CALL CHAR(108,"000003607387E1F0701C000000000003F000000000000080E0F87E1F0701000
0FF")
370 CALL CHAR(112,"0406070301000000C0C000000000003F000000080C0EC70381C0E070301000
0FF")
380 CALL CHAR(116,"0000000000000000000000000000003F8060301818180C0C0606030301000
0FF")
390 CALL CHAR(120,"000000000000C000000000000000003FC0C0C0C0C0606060603030303010000
0FF")
400 CALL CHAR(124,"60F1F3FFFFFFFFFFFF87F8FE8FF1FEFF")
410 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
420 CALL CHAR(140,"0000000000000103030100000000000000000000000080C0C0830000000000
000")
430 CALL HCHAR(24,14,124,19)
440 CALL HCHAR(24,1,125,13)
450 FOR I=1 TO 11
460 READ A,B,C
470 CALL HCHAR(A,B,C)
480 NEXT I
490 CALL SPRITE(#1,104,6,160,152)
500 FOR I=1 TO 7
510 READ A,B,C
520 CALL VCHAR(A,B,95,C)
530 NEXT I
540 FOR I=1 TO 14
550 READ A,B,C
560 CALL VCHAR(A,B,80,C)
570 NEXT I
580 FOR I=1 TO 18
590 READ A,B,C
600 CALL VCHAR(A,B,128,C)
610 NEXT I
620 FOR I=1 TO 36
630 READ A,B,C
640 CALL HCHAR(A,B,C)
650 NEXT I
660 RESTORE
670 GOTO 730
680 DATA 23,20,99,23,21,100,23,22,101,23,23,102,22,22,97,22,23,98,21,23,96,24,14
,73,24,15,75
690 DATA 23,14,72,23,15,74,20,25,4,18,26,6,19,27,5,21,28,3,18,29,6,20,30,4,19,31
,5
700 DATA 1,1,16,1,2,16,1,3,17,1,4,17,3,5,16,5,6,14,7,7,13,9,8,11,11,9,10,13,10,8
710 DATA 15,11,7,17,12,5,19,13,4,21,14,2,21,15,2,19,16,4,17,17,5,15,18,7,13,19,8
,11,20,10
720 DATA 9,21,11,7,22,10,5,23,14,3,24,16,1,25,17,1,26,17,1,27,16,1
730 DATA 28,16,1,29,15,1,30,15,1,31,14,1,32,14,23,1,50,23,2,51,22,1,48,22,2,49,2
1,1,57,21,2,58
```

```
740 DATA 20,1,56,20,2,58,23,11,57,23,12,59,22,11,56,22,12,58
750 DATA 23,3,57,23,4,59,22,3,56,22,4,58,23,5,50,23,6,51,22,5,48,22,6,49,21,5,57
,21,6,59
760 DATA 20,5,56,20,6,58,23,7,48,23,8,49,22,7,57,22,8,59,21,7,56,21,8,58
770 DATA 23,9,48,23,10,49,22,9,57,22,10,59,21,9,56,21,10,58
780 B=0 :: HA=33 :: TR=0
790 FOR I=1 TO 5 :: CALL SOUND(200,-3,0):: CALL SOUND(200,-1,0):: NEXT I
800 A=3 :: ZE=116 :: RANDOMIZE :: ZF=(INT(12*RND)+2)*8
810 CALL SPRITE(#3,64,2,ZF,1,0,28)
820 GOSUB 970 :: CALL POSITION(#3,Z1,S1):: IF S1>120 THEN CALL COLOR(7,2,1,13,11
,1)ELSE CALL COLOR(7,11,1,13,2,1)
830 CALL COINC(#2,#3,12,0):: IF 0=0 THEN 840 ELSE CALL DELSPRITE(#2,#3):: TR=TR+
1 :: GOTO 890
840 CALL KEY(1,K,S):: IF K<>18 OR B<>0 THEN 870 ELSE IF A=4 THEN ZL=142 :: SP=15
7 :: GH=-8 :: GV=-50 :: GOTO 850 ELSE IF A=3 THEN ZL=142 :: SP=149 :: GH=-20 ::
GV=-45 :: GOTO 860
850 IF A=2 THEN ZL=142 :: SP=139 :: GH=-33 :: GV=-37 :: GOTO 860 ELSE IF A=1 THE
N ZL=151 :: SP=134 :: GH=-40 :: GV=-22 :: GOTO 860 ELSE IF A=0 THEN 870
860 CALL SPRITE(#2,140,6,ZL,SP,GV,GH):: B=1 :: CALL SOUND(100,-6,0)
870 CALL POSITION(#2,Z1,S1):: IF Z1>16 AND Z1<176 AND S1>32 AND S1<158 THEN 880
ELSE CALL DELSPRITE(#2):: B=0
880 CALL COINC(#2,#3,12,0):: IF 0=0 THEN 900 ELSE CALL DE_SPRITE(#2,#3):: TR=TR+
1
890 FOR I=1 TO 5 :: FOR J=-3 TO -1 :: CALL SOUND(100,J,0):: NEXT J :: NEXT I ::
GOTO 900
900 CALL POSITION(#3,Z1,S1):: IF S1<160 THEN 820
910 RANDOMIZE :: CALL DELSPRITE(#2):: BO=(INT(7*RND))+25 :: GR=0 :: BM=0 :: B=0
920 CALL POSITION(#3,Z1,S1):: IF S1<(BO*8-8)OR BM=1 THEN 930 ELSE CALL SPRITE(#4
,68,11,110,(BO*8-10),9,0):: BM=1
930 IF GR=1 AND BM=0 THEN 810 ELSE GOSUB 970 :: CALL POSITION(#3,Z1,S1):: IF S1>
155 OR GR=1 THEN 940 ELSE CALL DELSPRITE(#3):: GR=1
940 IF BM=0 THEN 920 ELSE CALL POSITION(#4,Z2,S2):: IF Z2>192 THEN CALL DELSPRIT
E(#4):: GOTO 810 ELSE M=INT(Z2/8)+1 :: IF M>24 THEN M=24
950 CALL GCHAR(M,BC,T):: IF T<>95 THEN 920 ELSE CALL HCHAR(M,BO,32):: CALL DELSP
RITE(#4):: HA=HA-1
960 FOR I=-7 TO -5 :: CALL SOUND(200,I,0):: NEXT I :: IF HA=0 THEN 990 ELSE 810
970 CALL JOYST(1,X,Y):: IF Y=4 AND A<4 THEN ZE=ZE+4 :: A=A+1 ELSE IF Y=-4 AND A>
0 THEN ZE=ZE-4 :: A=A-1 ELSE 980
980 CALL PATTERN(#1,ZE):: RETURN
990 CALL SCREEN(8):: CALL DELSPRITE(ALL):: CALL CLEAR :: CALL CHARSET :: DISPLAY
 AT(8,3):"SIE HATTEN":TR;"ABSCHUESSE" :: DISPLAY AT(12,10):"NEUES SPIEL ? "
1000 DISPLAY AT(16,6):"JA >>(1)  NEIN >>(2)" :: ACCEPT AT(20,15)BEEP SIZE(1)VALI
DATE("12"):A :: IF A=1 THEN 200
1010 END
```

```
6930 PRINT "
"
6940 PRINT "
6950 PRINT AT 10,14;(" " AND X(I
*8-4)<10);X(I*8-4);AT 10,18;(" "
 AND X(I*8-3)<10);X(I*8-3);AT 14
,14;(" " AND X(I*8-2)<10);X(I*8-
2);AT 10,10;(" " AND X(I*8-1)<10
0);X(I*8-1)
6960 LET X=14
6970 LET Y=4
6980 RETURN
9000 SAVE "TES"
9010 GOTO 120

DIES IST EIN TEST,DER IHR
ZAHLENLOGISCHES VERSTAENDNIS
BEWERTEN SOLL.
ICH WERDE IHNEN 40 AUFGABEN
STELLEN,BEI DENEN SIE JEWEILS
DIE FEHLENDE,DURCH EIN FRAGE-
ZEICHEN ERSETZTE ZAHL EINGEBEN
SOLLEN.
SIE HABEN INSGESAMMT 30 MINUTEN
ZEIT.
BLEIBEN SIE DESHALB NICHT ZU
LANGE AN EINER AUFGABE HAENGEN,
MOEGLICHERWEISE SIND SIE AUF DER
FALSCHEN SPUR UND KOENNEN DIE
NAECHSTE AUFGABE BESSER LOESEN.
DIE UHR AM UNTEREN BILDRAND
ZEIGT IHNEN,WIEVIEL ZEIT SIE
NOCH HABEN.
DIE LOESUNG WIRD IMMER EINE ZAHL
VON 0 BIS 999 SEIN.
HABEN SIE EINE LOESUNG,GEBEN SIE
DIESE EIN UND DRUECKEN DANN AUF
DIE TASTE A.
HABEN SIE KEINE LOESUNG,DANN
GEBEN SIE NUR A EIN.
BLEIBT IHNEN AM SCHLUSS NOCH
ETWAS ZEIT,DANN KOENNEN SIE
DIESE AUFGABEN NOCHMALS PRO-
BIEREN.
WENN SIE DAS ZULETZT EINGEGEBENE
ZEICHEN LOESCHEN WOLLEN,DANN
DRUECKEN SIE AUF L.
```



Chopper-Command

für den TI-99/4A

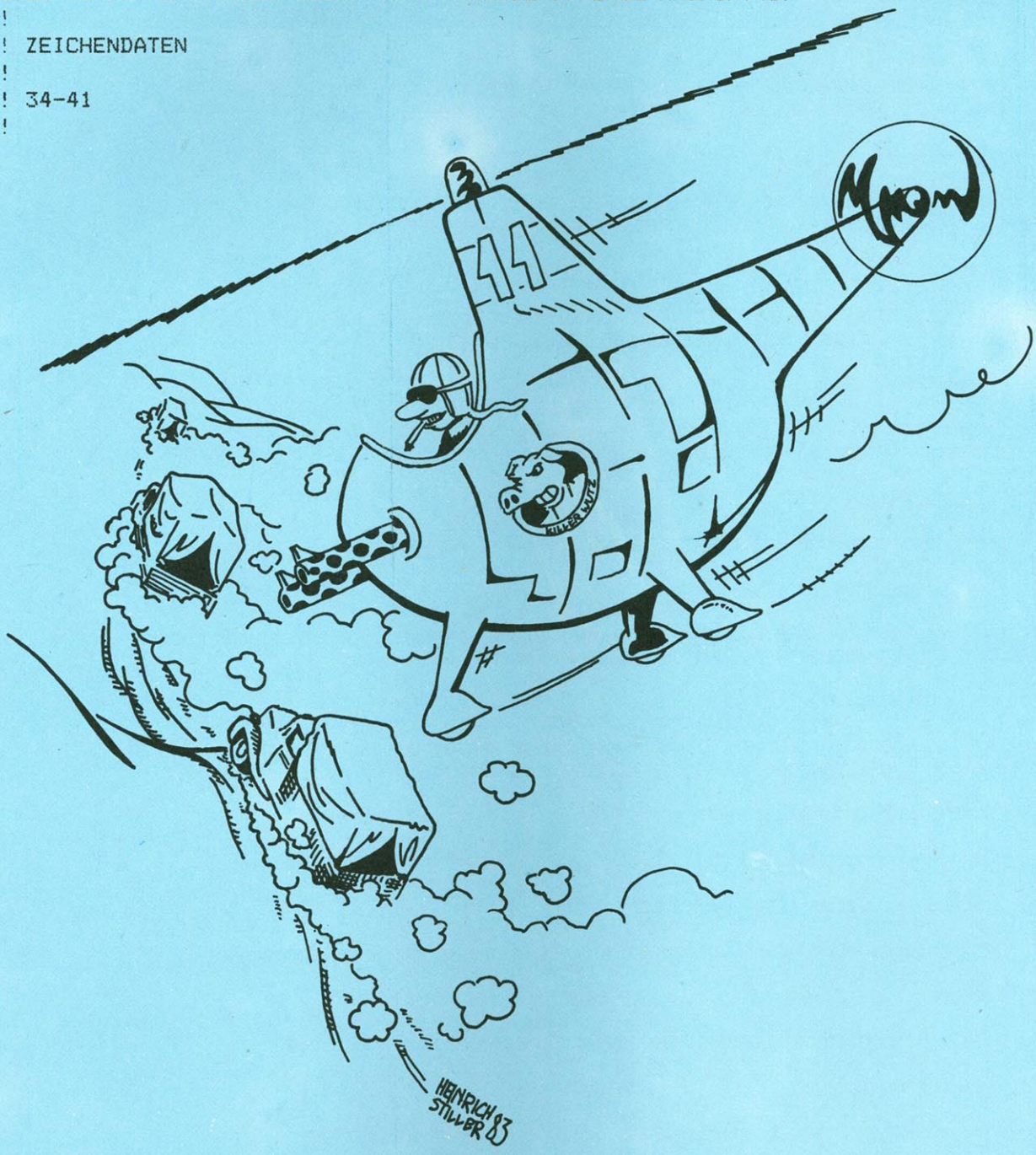
Ein Transporter-Treck ist unterwegs, um lebenswichtige Ladung, in ein kleines Dorf, in der Wüste Gobi, zu bringen.

Als Schutz gegen feindliche Kampfgeschwader, des in diesem Gebiet ansässigen Mongolenfürsten, werden Sie dem Transportunternehmen zugeteilt. Mit einem Hubschrauber begleiten Sie den Track und müssen ständig vor den Angreifern auf der Hut sein.
Mit der einzigen Schußwaffe an Bord, heißt es nun, soviel Treffer wie nur möglich zu erzielen, um die Zahl der LKW's, die von den feindlichen Jägern getroffen wurden und so letztendlich doch auf der Strecke blieben, möglichst gering zu halten.
Der Hubschrauber wird im Spiel mit dem Joystick gesteuert. Die Steuerung kann aber nur nach rechts, nach links und nach unten erfolgen, also nicht nach oben. Eine Tatsache, die das ganze Unternehmen noch schwieriger werden läßt.
Möchte man das Spiel über Tastatur spielen, sind folgende Zeilen einzufügen, bzw. zu ändern:

```
FALLS DAS SPIEL AUF DEM KEYBOARD GESPIELT WERDEN SOLL,
SIND FOLGENDE ZEILEN EINZUGEBEN bzw ABZUAENDERN !


   60 XJ(3) = 4
   70 XJ(2) = 4
   80 YJ(5) = 4
   90 YJ(0) = 4
 1560 CALL KEY(1,K,S):: GOSUB 1990 :: IF K>5 OR K<0 THEN K=4
 1565 ON K+1 GOTO 1590,1640,1580,1570,1590,1590
 1590 CALL MOTION(#1,-3*YJ(K),3*XJ(K))
 1640 CALL POSITION(#1,XP,YP) :: CALL SPRITE(#10,120,11,XP+1,YP+1,0,X+XJ(K))
      :: CALL SOUND(-200,-7,0)
 1680 GOTO 1590
```

```
100 ! CHOPPER COMMAND
110 !
120 ! BY T.BISPINK
130 !
140 ON ERROR 1560
150 CALL SCREEN(2):: CALL CLEAR :: RANDOMIZE :: CALL MAGNIFY(3)
160 !
170 ! ZEICHENDATEN
180 !
190 ! 34-41
200 !
```




```
210 DATA 6C6C6C,6CFEFE6C6CFEFE6C,6CFED0FC7E16FE6C,C6CE1E3C78F0E6C6,70F8D0767ECCF
E76,18182,1818303030301818,303018181818303
220 !
230 !42-49
240 !
250 DATA 0000001818,0018187E7E1818,000000000018182,0000007E7E,0000000000001818,0
60E1E3C78F0E0C,7CFECEDEF6E6FE7C,1838787818181818
260 !
```

```
270 !50-56
280 !
290 DATA 7CFE0E3C78E0FEFE,7CFE0E3C3C0EFE7C,1C3C7CECFEFE0C0C,FEFEC0FCFE06FE7C,7CF
EC0FCFEC6FE7C,FEFE060E1C38303,7CFEC67C7CC6FE7C
300 !
310 !57-64
320 !
330 DATA 7CFEC6FE7E06FE7C,00181800001818,001818000018182,061E7CF0F07C1E06,007C7C
00007C7C,C0F07C1E1E7CF0C,7CFE061E1C001818,C6CEFC783030303
340 !
350 ! 65-72
360 !
370 DATA 387CEEC6FEFEC6C6,FCFEC6FCFCC6FEFC,7CFEC0C0C0C0FE7C,FCFEC6C6C6C6FEFC,FEF
EC0F8F8C0FEFE,FEFEC0F8F8C0C0C0,7CFEC0CECEC6FE7C,C6C6C6FEFEC6C6C6
380 !
390 ! 73-80
400 !
410 DATA 1818181818181818,FEFE06060606FE7C,C6CEDCF8F8DCCEC6,C0C0C0C0C0C0FEFE,C6E
EFEFED6C6C6C6,C6E6E6F6DECECEC6,7CFEC6C6C6C6FE7C,FCFEC6FEFCC0C0C0
420 !
430 ! 81-87
440 !
450 DATA C6EE7C38387CEEC6,FCFEC6FEFCDCCEC6,7CFEC0FC7E06FE7C,FEFE181818181818,C6C
6C6C6C6C6FE7C,C6C6EE6C7C38381,C6C6C6C6D6FEFE6C
460 !
470 ! HUBSCHRAUBER 92-95,96-99
480 !
490 DATA 000000070040E17F03000000000000000000000FF207EF1F9FEF8127C000000000,00000
0FF047E8F9F7F1F483E00000000000000E0000287FEC00000000000000
500 !
510 ! LKW 100-103
520 !
530 DATA 00000000000031F2747FFFF7F57381000000000000FFFFFFFFFFFFFFEA1C08
540 !
550 ! FLUGZEUGE 104-107,108-111
560 !
570 DATA 0000000000001FE7F83F0700000000000000000000010306FE0EF8E00000000000,000000
0080C0607F701F0700000000000000000000000F8E71FFCE00000000000
580 !
590 ! STARFIGHTER 112-115,116-119
600 !
610 DATA 0000000001073FE73F07010000000000000000F3FF8E0FFC0FFE0F83F0F000000,0000F0
FC1F07FF03FF070FFCF0000000000000000080E0FCE7FCE0800000000000
620 !
630 !SCHUSS HUBSCHRAUBER 120-123
640 !
650 DATA 000000000000FF0000FF000000000000000000000000000FF0000FF000000000000
660 !
670 ! SCHUSS FLUGZEUG 124-         127
680 !
690 DATA 00000000000000030300000000000000000000000000000C0C000000000000000
700 !
710 ! BOMBE  128-131
720 !
730 DATA 0000000000007EC37E0000000000000000000000000000000000000000000000000
740 !
750 ! EXPLOSION 132-135
760 !
770 DATA 80603C1F0F1F7C3B0B0D0E1F1930200040E0E0F8FF7EB8D8B8DE59B8FCFC8E06
780 !
790 ! GEBIRGE 136-140
800 !
810 DATA 00000000030F3FFF,010F3FFFFFFFFFF,00C0F0FCFFFFFFFF,000000080F0FCFE,FFF
FFFFFFFFFFF
820 !
830 ! HIMMEL 141
840 !
850 DATA 0000000000000000
860 !
870 ! WUESTE 33
```

```
880 !
890 DATA FFFFFFFFFFFFFFFF
900 !
910 ! GEBIRGEPOSITION
920 !
930 DATA 7,1,6,3,6,7,5,9,4,11,5,17,6,21,5,23,4,25,3,27
940 DATA 7,2,6,4,6,8,5,10,4,12,6,22,5,24,4,26,3,28
950 DATA 6,5,4,13,5,15,6,19,3,29,4,31
960 DATA 6,6,4,14,5,16,5,18,6,20,3,30,4,32
970 !
980 ! ZEICHENDEFINITIONEN
990 !
1000 FOR I=34 TO 87 :: READ A$ :: CALL CHAR(I,A$):: NEXT I
1010 FOR I=92 TO 132 STEP 4 :: READ A$ :: CALL CHAR(I,A$):: NEXT I
1020 FOR I=136 TO 140 :: READ A$ :: CALL CHAR(I,A$):: NEXT I
1030 READ A$ :: CALL CHAR(141,A$):: READ A$ :: CALL CHAR(33,A$)
1040 !
1050 ! FARBEN
1060 !
1070 CALL COLOR(1,7,7):: CALL COLOR(14,14,8):: FOR I=2 TO 7 :: CALL COLOR(I,11,7
):: NEXT I
1080 !
1090 ! BILD
1100 !
1110 CALL HCHAR(1,1,141,224):: CALL HCHAR(8,1,33,544)
1120 CALL HCHAR(6,9,140,24):: CALL HCHAR(7,3,140,30):: CALL HCHAR(5,11,140,4)::
CALL HCHAR(5,25,140,8):: CALL HCHAR(4,27,140,4)
1130 FOR I=1 TO 10 :: READ A,B :: CALL HCHAR(A,B,136):: NEXT I
1140 FOR I=1 TO 9 :: READ A,B :: CALL HCHAR(A,B,137):: NEXT I
1150 FOR I=1 TO 6 :: READ A,B :: CALL HCHAR(A,B,138):: NEXT I
1160 FOR I=1 TO 7 :: READ A,B :: CALL HCHAR(A,B,139):: NEXT I
1170 !
1180 ! TITELSCHRIFT
1190 !
1200 DISPLAY AT(12,8):"CHOPPER COMMAND" :: DISPLAY AT(16,6):"1 ODER 2 SPIELER ?"
1210 CALL SPRITE(#4,100,2,175,176,0,-6,#5,100,2,175,216,0,-6,#6,100,2,175,256,0,
-6)
1220 CALL SPRITE(#1,92,5,40,1,0,8)
1230 CALL KEY(0,K,S):: IF S=0 OR K>50 OR K<49 THEN 1230
1240 A=K-48 :: CALL HCHAR(10,1,33,480):: FOR I=2 TO 7 :: CALL COLOR(I,5,8):: NEX
T I :: CALL COLOR(1,7,8)
1250 !
1260 ! SPIELRUNDE
1270 !
1280 P=P+1 :: IF P>A THEN 1290 ELSE 1300
1290 P=0 :: R=R+1 :: IF R>2 THEN 1440 ELSE 1280
1300 X=100 :: L(1)=0 :: L(2)=0 :: CALL DELSPRITE(ALL)
1310 FOR I=1 TO 2-R :: CALL SPRITE(#I+1,92,5,1,(I-1)*18+9):: NEXT I
1320 DISPLAY AT(2,5):"SPIELER:";P :: DISPLAY AT(2,17):USING "#####":T(P):: DISPL
AY AT(2,23):"PUNKTE"
1330 FOR I=4 TO 6 :: CALL SPRITE(#I,100,2,175,256,0,-6):: FOR J=1 TO 400 :: NEXT
 J :: NEXT I
1340 CALL SPRITE(#1,92,16,100,128)
1350 ON INT(RND*4)+1 GOTO 1720,1760,1840,1880
1360 T(P)=T(P)+N :: DISPLAY AT(2,17)SIZE(5):USING "#####":T(P):: GOTO 1350
1370 T(P)=T(P)-100 :: DISPLAY AT(2,17)SIZE(5):USING "#####":T(P):: L(P)=L(P)+1 :
: IF L(P)=3 THEN 1380 ELSE 1350
1380 IF T(P)>1999 AND F(P)=0 THEN F(P)=1 ELSE GOTO 1280
1390 FOR I=1 TO 5 :: DISPLAY AT(2,1):"      EQTENDED PLA$" :: CALL SOUND(70,1300
,0):: FOR J=1 TO 100 :: NEXT J
1400 DISPLAY AT(2,1):"" :: FOR J=1 TO 50 :: NEXT J :: NEXT I :: GOTO 1300
1410 !
1420 ! ANZEIGE ENDERGEBNIS
1430 !
1440 CALL HCHAR(2,1,141,32):: CALL COLOR(1,7,7):: FOR I=2 TO 7 :: CALL COLOR(I,1
1,7):: NEXT I
1450 DISPLAY AT(10,3):"SPIELER 1:";T(1);"PUNKTE" :: IF A=1 THEN 1470
1460 DISPLAY AT(12,3):"SPIELER 2:";T(2);"PUNKTE"
1470 DISPLAY AT(15,5):"NEUES SPIEL?" :: DISPLAY AT(17,3):"1 ODER 2 SPIELER ?"
1480 FOR J=1 TO 7 :: CALL KEY(0,K,S):: IF K=49 OR K=50 THEN 1520
```

```
1490 DISPLAY AT(22,10):"GAME OVER" :: NEXT J
1500 FOR J=1 TO 8 :: CALL KEY(0,K,S):: IF K=49 OR K=50 THEN 1520
1510 DISPLAY AT(22,10):"" :: NEXT J :: GOTO 1480
1520 F(1),F(2),T(1),T(2),P,R=0 :: GOTO 1240
1530 !
1540 ! UNTERPROGRAMM STEUERUNG HUBSCHRAUBER
1550 !
1560 CALL KEY(P,K,S):: GOSUB 1990 :: CALL JOYST(P,XJ,YJ):: ON XJ/4+2 GOTO 1580,1
590,1570
1570 CALL PATTERN(#1,92):: X=100 :: GOTO 1590
1580 CALL PATTERN(#1,96):: X=-100
1590 CALL MOTION(#1,-3*YJ,3*XJ):: IF K=18 THEN GOSUB 1640
1600 ON INT(RND*20)+1 GOTO 1560,1920,1560,1560,1920,1560,1800,1560,1920,1560,135
0,1560,1560,1560,1800,1560,1560,1560,1920,1560
1610 !
1620 ! UNTERPROGRAMM SCHUSS HUBSCHRAUBER
1630 !
1640 CALL POSITION(#1,XP,YP):: CALL SPRITE(#10,120,11,XP+1,YP+1,0,X+XJ):: CALL S
OUND(-200,-7,0)
1650 CALL POSITION(#7,XF,YF):: IF ABS(XP-XF)>5 OR(YF-YP)*X<0 THEN 1670 :: CALL D
ELSPRITE(#7,#10):: CALL MOTION(#1,0,0)
1660 CALL SPRITE(#13,132,11,XF,YF,0,0):: CALL SOUND(700,-6,0):: FOR J=1 TO 200 :
: NEXT J :: CALL DELSPRITE(#13):: GOTO 1360
1670 CALL DELSPRITE(#10)
1680 RETURN
1690 !
1700 ! BEWEGUNG FLUGZEUG VON RECHTS
1710 !
1720 N=50+R*20 :: G=-1 :: CALL DELSPRITE(#7):: CALL SPRITE(#7,104,2,60+INT(80*RN
D),256,0,-25-8*R):: GOTO 1560
1730 !
1740 ! BEWEGUNG FLUGZEUG VON LINKS
1750 !
1760 N=50+R*20 :: G=1 :: CALL DELSPRITE(#7):: CALL SPRITE(#7,108,2,60+INT(80*RND
),1,0,25+8*R):: GOTO 1560
1770 !
1780 ! BOMBE FLUGZEUG
1790 !
1800 CALL POSITION(#7,XF,YF):: CALL SPRITE(#12,128,2,XF,YF+1,10,20*G):: CALL SOU
ND(-50,120,8):: GOTO 1560
1810 !
1820 ! BEWEGUNG STARFIGHTER VON RECHTS
1830 !
1840 N=100+R*30 :: G=-1 :: CALL DELSPRITE(#7):: CALL SPRITE(#7,112,2,40+INT(110*
RND),256,0,-40-10*R):: GOTO 1560
1850 !
1860 ! BEWEGUNG STARFIGHTER VON LINKS
1870 !
1880 N=100+R*30 :: G=1 :: CALL DELSPRITE(#7):: CALL SPRITE(#7,116,2,40+INT(110*R
ND),1,0,40+10*R):: GOTO 1560
1890 !
1900 ! SCHUSS STARFIGHTER
1910 !
1920 CALL POSITION(#7,XF,YF):: CALL MOTION(#11,0,120*G):: CALL SPRITE(#11,124,2,
XF,YF+1):: CALL SOUND(100,-5,0)
1930 CALL POSITION(#1,XP,YP):: IF ABS(XP-XF)>5 OR(YF-YP)*G>0 THEN 1950 :: CALL D
ELSPRITE(#1,#11)
1940 CALL SPRITE(#13,132,10,XP,YP,0,0):: CALL SOUND(700,-6,0):: FOR J=1 TO 200 :
: NEXT J :: CALL DELSPRITE(#13):: GOTO 1380
1950 CALL DELSPRITE(#11):: GOTO 1560
1960 !
1970 ! BOMBEN AUF WAGENKOLONNE
1980 !
1990 CALL POSITION(#12,XB,YB):: IF XB<165 THEN RETURN
2000 FOR I=4 TO 6 :: CALL COINC(#12,#I,22,C(I)):: IF C(I)=-1 THEN 2020
2010 NEXT I :: CALL DELSPRITE(#12):: RETURN
2020 CALL MOTION(#1,0,0):: CALL POSITION(#I,XB,YB):: CALL SPRITE(#I,100,8,1,1,0,
0):: CALL DELSPRITE(#12):: IF YB>250 THEN YB=250
2030 CALL SPRITE(#13,132,10,175,YB+1,0,0):: CALL SOUND(700,-6,0):: FOR J=1 TO 20
0 :: NEXT J :: CALL DELSPRITE(#13):: GOTO 1370
```

LABYRINTH ZICK-ZACK

für den TI-99/4A

Für dieses Programm wird das Ext. Basic-Modul und der Joystick benötigt.

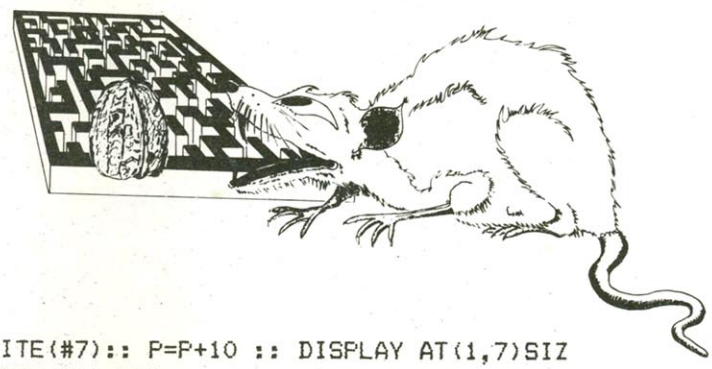
Der Bildschirmaufbau besteht aus einem Labyrinth aus hartem Gestein, durch das Sie sich als kleine gefräßige Maus durchbeißen müssen, um die Nüsse vertilgen zu können, die auf den Wegen im Labyrinth ausliegen.

Da das Durchbeißen des Gesteins natürlich sehr viel Kraft verbraucht, ist es von Zeit zu Zeit erforderlich, einen Spaten zur Hilfe zu nehmen, um weitergraben zu können.
Der Kreislauf schließt sich damit, daß Ihnen ebenfalls eine Spinne auf den Versen ist die Sie, in der Rolle als Maus, vertilgen will.
Die Devise lautet also wiedereinmal: Fressen oder gefressen werden.

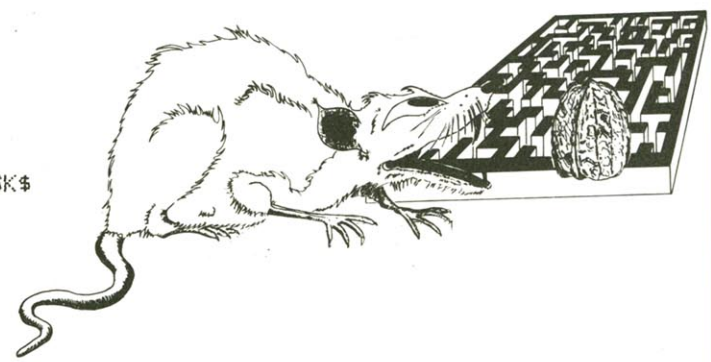
```
100 ON BREAK NEXT :: RANDOMIZE :: HISC=0
110 !
120 ! ZICK ZACK
130 ! BY C.$TELLING
140 ! (C) 27.11.1983
150 !
160 CALL CLEAR :: CALL SCREEN(5):: DISPLAY AT(12,12)BEEP:"ICH" :: DISPLAY AT(14,
12):"RECHNE"
170 CALL COLOR(0,16,1):: FOR CO=2 TO 9 :: CALL COLOR(CO,16,1):: NEXT CO
180 !
190 ! ZEICHENDATAS
200 !
210 DATA 0000003030102040,,0000000000001818,
220 DATA 3C7E666666667E3C,183878181818183C,7E7E067E7E607E7E,7E7E067E7E067E7E,0C1
C3C6C6E7E0C0C
230 DATA 7E7E607E7E067E7E,7E7E607E7E667E7E,7E7E0C0C18183030,3C7E667E7E667E3C,7E7
E667E7E067E7E
240 DATA 0018180000181800,,,,,,
250 !
260 ! BUCHSTABEN
270 !
```

```
280 DATA 7E7E667E7E666666,7C7E667C7C667E7C,3E7E606060607E3E,7C7E666666667E7C,7E7
E607E7E607E7E
290 DATA 7E7E607C7C606060,3E7E607E7E667E3C,6666667E7E666666,1818181818181818,060
6060666667E3C
300 DATA 666E7C78787C6E66,6060606060607E7E,42667E7E66666666,86C6E6F6DECEC6C2,7E7
E666666667E7E
310 DATA 7C7E667E7C606060,3C7E66666E667E3F,7C7E667E7C6C6666,3E7E607C3E067E7C,7E7
E181818181818
320 DATA 6666666666667E7E,6666666663C3C18,C6C6C6C6D6FEEEC6,66663C18183C6666,81C
3663C18181818
330 DATA 7E7E060C18307E7E
340 !
350 ! FIGUREN - DATAS
360 !
370 DATA 000103050F07171F0F0717170E01011C0080C0A0F0E0E8F8F0E0E8E8D0000000
380 DATA 0000000E1F312003060D0B0B0B05030000464E9CB8FEC0C0E0F0F0F0E0C08000
390 DATA 0000020402314B47073F47031F2341000000402040BCD2E0E0FCE2C0F8C48200
400 DATA 0000001818000000
410 !
420 ! ERDE WEG
430 !
440 DATA FFFFFFFFFFFFFFFF
450 !
460 ! SPATEN
470 !
480 DATA 00000000000103277EFCFCFEFFFEFC0000183870E0C080000000000000000000
490 !
500 ! FELSBROCKEN
510 !
520 DATA 00070E0D1B3B777777371717130904030080F0F0F8FCF8FCFEFCFCF8F8F8F0E0
530 DATA 0000000000000103030100000000000000000000000080C0C080000000000000
540 !
550 ! CHAR - DEFINIERUNG
560 !
570 RESTORE :: FOR A=44 TO 90 :: READ B$ :: CALL CHAR(A,B$):: NEXT A
580 RESTORE 520 :: READ B$ :: CALL CHAR(112,B$)
590 RESTORE 370 :: READ C$ :: CALL CHAR(128,C$)
600 RESTORE 380 :: READ D$ :: CALL CHAR(96,D$)
610 RESTORE 390 :: READ D$ :: CALL CHAR(100,D$)
620 RESTORE 440 :: READ E$ :: CALL CHAR(33,E$)
630 RESTORE 400 :: READ E$ :: CALL CHAR(40,E$)
640 RESTORE 480 :: READ E$ :: CALL CHAR(104,E$)
650 RESTORE 530 :: READ F$ :: CALL CHAR(120,F$)
660 GOSUB 1780
670 CALL CLEAR
680 CALL COLOR(1,2,1)
690 !
700 ! HAUPTPROGRAMM
710 !
720 ! ZEICHNEN
730 !
740 K=0
750 P=0
760 F=3
770 T=100
780 SPEED=10
790 Y10=RND*20+40
800 X10=RND*184+40
810 !
820 CALL SPRITE(#1,128,16,32,120)
830 CALL SPRITE(#6,100,4,88,16)
840 CALL SPRITE(#3,100,8,88,104)
850 CALL SPRITE(#4,100,14,88,184)
860 CALL SPRITE(#5,96,10,RND*170+8,RND*232+8)
```

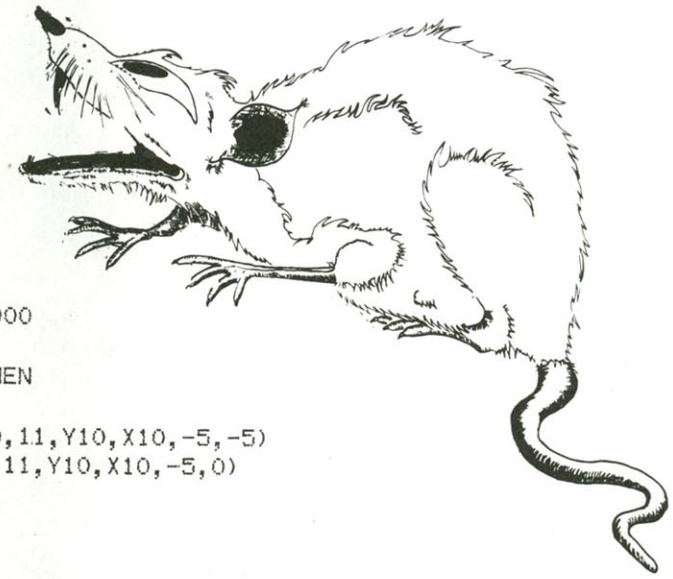
```
870 CALL SPRITE(#7,104,8,RND*50+8,RND*232+8)
880 CALL SPRITE(#10,112,12,Y10,X10)
890 CALL MAGNIFY(3)
900 CALL HCHAR(12,1,33,32)
910 CALL HCHAR(13,1,33,32)
920 CALL HCHAR(4,14,33,6):: CALL HCHAR(5,14,33,6):: CALL HCHAR(6,14,33,6):: CALL
 HCHAR(7,14,33,6)
930 DISPLAY AT(1,1):"SCORE :";P :: DISPLAY AT(1,15):"HISCORE :";HISC
940 !
950 X1=120 :: Y1=32
960 X2=8 :: Y2=88
970 !
980 ! JOYST
990 !
1000 CALL JOYST(2,X,Y)
1010 T=T-1
1020 IF P>5000 THEN SPEED=SPEED+.1 :: IF SPEED>40 THEN SPEED=39
1030 IF X=-4 THEN 1040 ELSE IF X=4 THEN 1060 ELSE IF Y=4 THEN 1080 ELSE IF Y=-4
THEN 1100 ELSE 1460
1040 ! NACH LINKS
1050 X1=X1-8 :: CALL LOCATE(#1,Y1,X1):: GOTO 1120
1060 ! NACH RECHTS
1070 X1=X1+8 :: CALL LOCATE(#1,Y1,X1):: GOTO 1120
1080 ! NACH OBEN
1090 Y1=Y1-8 :: CALL LOCATE(#1,Y1,X1):: GOTO 1120
1100 ! NACH UNTEN
1110 Y1=Y1+8 :: CALL LOCATE(#1,Y1,X1):: GOTO 1120
1120 IF X1<16 OR X1>240 OR Y1<16 OR Y1>172 THEN 1130 ELSE 1150
1130 CALL LOCATE(#1,32,120)
1140 Y1=32 :: X1=120
1150 CALL GCHAR(Y1/8+1,X1/8+1,Z)
1160 IF Z=32 THEN 1310
1170 CALL POSITION(#1,Y1,X1)
1180 CALL COINC(#1,#5,8,Z2)
1190 CALL COINC(#1,#7,8,Z3)
1200 IF Z2=-1 THEN 1370
1210 IF Z3=-1 THEN 1260
1220 GOTO 1470
1230 !
1240 ! SPATEN HOLEN
1250 !
1260 CALL SOUND(-300,1000,0):: CALL DELSPRITE(#7):: P=P+10 :: DISPLAY AT(1,7)SIZ
E(7):P :: CALL SPRITE(#7,104,8,RND*152+24,RND*216+32)
1270 T=100 :: GOTO 1480
1280 !
1290 ! GRABEN
1300 !
1310 IF T<=0 THEN 1650 :: CALL SOUND(-40,-5,5)
1320 CALL HCHAR(Y1/8+1,X1/8+1,33,2):: CALL HCHAR(Y1/8+2,X1/8+1,33,2)
1330 P=P+10
1340 DISPLAY AT(1,7)SIZE(7):P :: GOTO 1480
1350 !
1360 ! NUSS FRESSEN
1370 !
1380 FOR I=1 TO 8 :: CALL SOUND(10,-7,0):: FOR I2=1 TO 10 :: NEXT I2 :: NEXT I :
: CALL SOUND(-45,-6,0)
1390 CALL DELSPRITE(#5)
1400 CALL SPRITE(#5,96,10,((RND*21)+1)*8,((RND*30)+1)*8)
1410 K=K+10
1420 P=P+K
1430 DISPLAY AT(1,7)SIZE(7):P
1440 GOTO 1480
1450 !
```



```
1460 ! SPINNE (SPIDER)
1470 !
1480 X3=SGN(X1-X2)
1490 Y3=SGN(Y1-Y2)
1500 IF X3=-1 THEN 1510 ELSE IF X3=1 THEN 1520 ELSE IF Y3=-1 THEN 1530 ELSE IF Y
3=1 THEN 1540 ELSE 1000
1510 X2=X2-8 :: GOTO 1550
1520 X2=X2+8 :: GOTO 1550
1530 Y2=Y2-8 :: GOTO 1550
1540 Y2=Y2+8
1550 CALL LOCATE(#6,Y2,X2)
1560 CALL COINC(#1,#6,8,B):: IF B=-1 THEN 1650
1570 CALL MOTION(#3,0,SPEED):: CALL MOTION(#4,0,-SPEED)
1580 CALL COINC(#1,#3,8,D):: CALL COINC(#1,#4,8,E)
1590 IF D=-1 THEN 1650
1600 IF E=-1 THEN 1650
1610 CALL GCHAR(Y10/8+2,X10/8+1,G):: IF G<>32 THEN 2220 ELSE 1000
1620 !
1630 ! MAUS GEFRESSEN
1640 !
1650 FOR I3=550 TO 110 STEP -10 :: CALL SOUND(-100,I3,0):: NEXT I3
1660 CALL DELSPRITE(ALL)
1670 DISPLAY AT(4,10):"ZICK ZACK"
1680 IF P>HISC THEN 1690 ELSE 1710
1690 HISC=P :: DISPLAY AT(1,23):;HISC
1700 DISPLAY AT(6,12)SIZE(7):"RECORD"
1710 DISPLAY AT(10,3):"NUR MUT" :: DISPLAY AT(14,3):"WOLLEN SIE NOCHMAL" :: DISP
LAY AT(16,3):"ZICK ZACK SPIELEN"
1720 DISPLAY AT(20,3):"DRUECKEN SIE :              "
1730 DISPLAY AT(24,3):"BEGIN ODER BACK             "
1740 CALL KEY(0,K,S):: IF S=0 THEN 1740
1750 IF K=14 THEN 670 :: IF K=15 THEN 100 ELSE 1740
1760 !
1770 ! SPIEL BEKANNT ?
1780 !
1790 CALL CLEAR
1800 DISPLAY AT(1,10):"ZICK ZACK"
1810 DISPLAY AT(12,1):"KENNEN SIE ZICK ZACK"
1820 ACCEPT AT(12,23)VALIDATE(UALPHA)SIZE(1):ASK$
1830 IF ASK$="J" THEN RETURN
1840 IF ASK$="N" THEN 1850
1850 CALL CLEAR
1860 DISPLAY AT(1,10):"ZICK ZACK"
1870 DISPLAY AT(4,1):"ZICK ZACK WURDE VON"
1880 DISPLAY AT(6,1):"CARSTEN STELLING AM"
1890 DISPLAY AT(8,1):"27.11.1983 PROGRAMMIERT."
1900 DISPLAY AT(12,1):"ALLE RECHTE VORBEHALTEN."
1910 DISPLAY AT(20,1):"BITTE ENTER DRUECKEN"
1920 CALL KEY(0,K,S):: IF S=0 THEN 1920 :: IF K=13 THEN 1930 ELSE 1920
1930 CALL CLEAR
1940 DISPLAY AT(1,10):"ZICK ZACK"
1950 DISPLAY AT(4,1):"SIE MUESSEN VERSUCHEN MIT"
1960 DISPLAY AT(6,1):"IHRER MAUS VOR DER SPINNE"
1970 DISPLAY AT(8,1):"ZU FLUECHTEN. SIE KOENNEN"
1980 DISPLAY AT(10,1):"NUR EINE BESTIMMTE ZEIT    "
1990 DISPLAY AT(12,1):"GRABEN OHNE EINEN NEUEN    "
2000 DISPLAY AT(14,1):"SPATEN GEHOLT ZU HABEN     "
2010 DISPLAY AT(16,1):"SIE KOENNEN ABER DIE       "
2020 DISPLAY AT(18,1):"SPINNE UEBERWAELTIGEN,     "
2030 DISPLAY AT(20,1):"INDEM SIE IHR EINEN        "
2040 DISPLAY AT(22,1):"FELSBROCKEN UEBERWERFEN.   "
2050 DISPLAY AT(24,1):"BITTE ENTER DRUECKEN"
2060 CALL KEY(0,K,S):: IF S=0 THEN 2060 :: IF K=13 THEN 2070 ELSE 2060
2070 CALL CLEAR
```

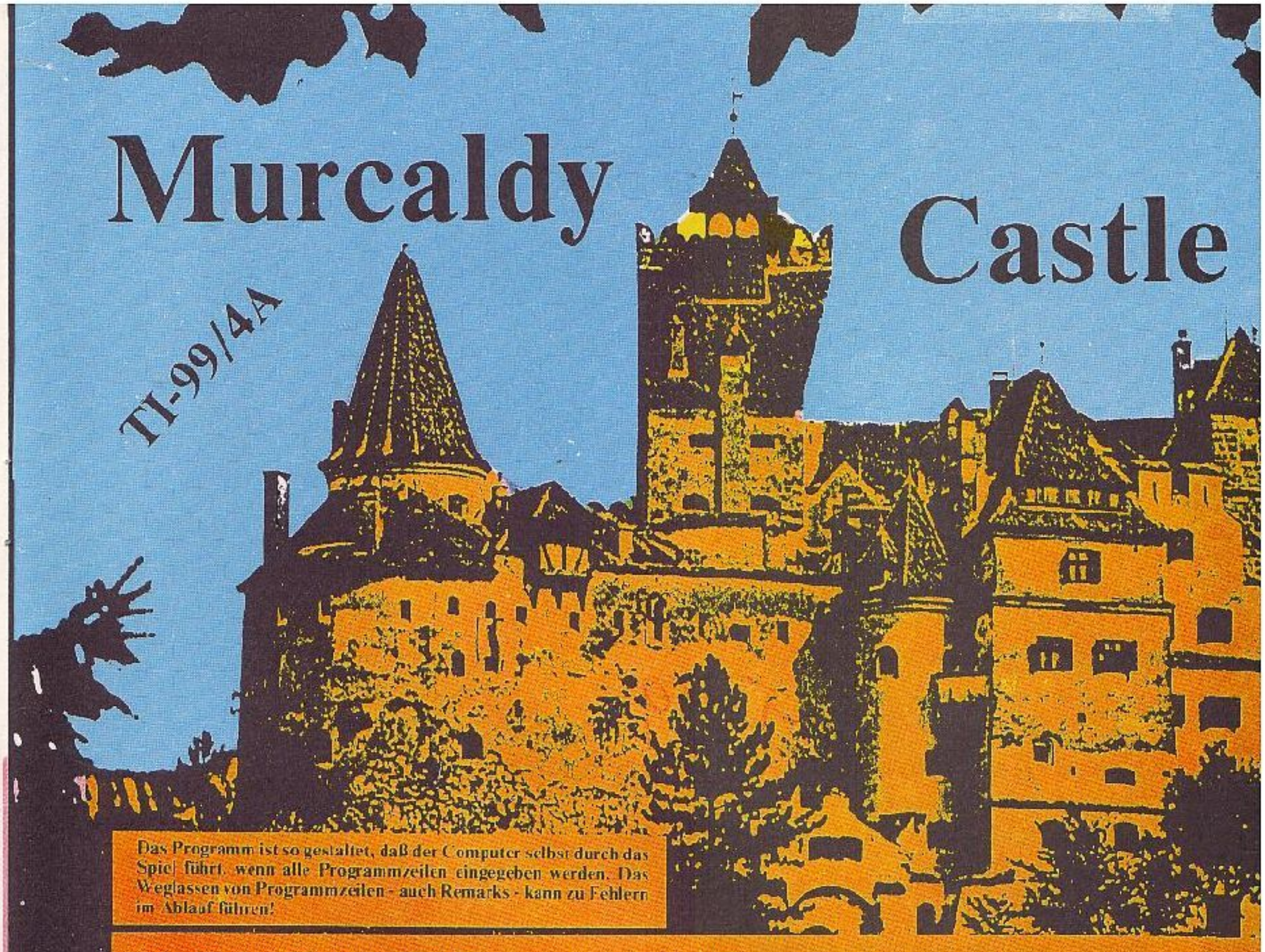


```
2080 DISPLAY AT(1,10):"ZICK ZACK"
2090 DISPLAY AT(4,1):"DAS GELINGT, INDEM SIE    "
2100 DISPLAY AT(6,1):"DIE STELLE UNTER DEM      "
2110 DISPLAY AT(8,1):"FELSBROCKEN UNTERHOEHLEN. "
2120 DISPLAY AT(10,1):"ZU GUTER LETZT KOENNEN   "
2130 DISPLAY AT(12,1):"SIE PUNKTE SAMMELN INDEN "
2140 DISPLAY AT(14,1):"SIE DIE ROTEN NUESSE     "
2150 DISPLAY AT(16,1):"      FRESSEN.           "
2160 DISPLAY AT(20,1):"VIEL GLUECK ...          "
2170 DISPLAY AT(24,1):"BITTE BEGIN DRUECKEN     "
2180 CALL KEY(0,K,S):: IF S=0 THEN 2180 :: IF K=14 THEN RETURN ELSE 2180
2190 !
2200 ! FELSBROCKEN FAELLT
2210 !
2220 FOR I6=1 TO 8 :: CALL SOUND(-200,-6,28):: FOR I7=-1 TO 1 STEP 1 :: CALL LOC
ATE(#10,Y10,X10+I7):: NEXT I7 :: NEXT I6
2230 S0=190
2240 Y10=Y10+8 :: X10=X10+0 :: CALL LOCATE(#10,Y10,X10)
2250 CALL COINC(#10,#6,8,Z5)
2260 CALL COINC(#10,#4,8,Z6)
2270 CALL COINC(#10,#3,8,Z7)
2280 IF Z5=-1 THEN 2350
2290 IF Z6=-1 THEN 2360
2300 IF Z7=-1 THEN 2370
2310 S0=S0-4 :: CALL SOUND(-300,S0,5,S0+2,5,S0+4,5)
2320 CALL GCHAR(Y10/8+2,X10/8,Z8):: IF Z8<>33 THEN 2330 ELSE 2240
2330 F=F-1 :: IF F=0 THEN 2340 ELSE 2530
2340 CALL DELSPRITE(#10):: Y10=1 :: X10=1 :: GOTO 1000
2350 CALL DELSPRITE(#6):: GOTO 2410
2360 CALL DELSPRITE(#4):: GOTO 2410
2370 CALL DELSPRITE(#3):: GOTO 2410
2380 !
2390 ! FELS TRIFFT SPINNE
2400 !
2410 CALL SOUND(-2000,-7,0)
2420 P=P+1000 :: DISPLAY AT(1,7)SIZE(7):P
2430 CALL DELSPRITE(#10)
2440 Y10=RND*20+40 :: X10=RND*184+40
2450 CALL SPRITE(#10,112,11,Y10,X10)
2460 CALL HCHAR(Y10/8-1,X10/8-1,32,4)
2470 CALL HCHAR(Y10/8,X10/8-1,32,4)
2480 CALL HCHAR(Y10/8+1,X10/8-1,32,4)
2490 CALL HCHAR(Y10/8+2,X10/8-1,32,4):: GOTO 1000
2500 !
2510 ! FELS TRIFFT ERDE               NEU ZEICHNEN
2520 !
2530 CALL SOUND(500,-5,0):: CALL SPRITE(#20,120,11,Y10,X10,-5,-5)
2540 CALL DELSPRITE(#10):: CALL SPRITE(#21,120,11,Y10,X10,-5,0)
2550 CALL SPRITE(#22,120,11,Y10,X10,-5,5)
2560 CALL SPRITE(#23,120,11,Y10,X10,0,-5)
2570 CALL SPRITE(#24,120,11,Y10,X10,0,5)
2580 CALL SPRITE(#25,120,11,Y10,X10,5,-5)
2590 CALL SPRITE(#26,120,11,Y10,X10,5,0)
2600 CALL SPRITE(#27,120,11,Y10,X10,5,5)
2610 FOR I4=1 TO 150 :: NEXT I4 :: CALL DELSPRITE(#20,#21,#22,#23,#24,#25,#26,#2
7)
2620 CALL DELSPRITE(#10)
2630 Y10=RND*20+40 :: X10=RND*184+40
2640 CALL SPRITE(#10,112,11,Y10,X10)
2650 CALL HCHAR(Y10/8,X10/8-1,32,4)
2660 CALL HCHAR(Y10/8+1,X10/8-1,32,4)
2670 CALL HCHAR(Y10/8+2,X10/8-1,32,4)
2680 CALL HCHAR(Y10/8+3,X10/8-1,32,4):: GOTO 1000
```



Murcaldy Castle

TI-99/4A



Das Programm ist so gestaltet, daß der Computer selbst durch das Spiel führt, wenn alle Programmzeilen eingegeben werden. Das Weglassen von Programmzeilen - auch Remarks - kann zu Fehlern im Ablauf führen!

Nach dem Start mit RUN erscheint nach wenigen Sekunden (blau auf grün) das Haupttitelbild mit den Wahlmöglichkeiten (1) Spielstart und (2) Spielregel. Ein Unterbrechen des Programms ist jetzt nicht mehr möglich (bzw. nur durch QUIT), es sei denn, die Option ON BREAK NEXT wurde ausgelassen.

Das Spiel: Malcolm MC Money - dargestellt als lustiger, blauer Kopf eines Männchens (bleu), soll aus der Burg Murcaldy einen Schatz bergen. Hindernisse dabei sind Bodenkontakte, die Türen öffnen und schließen sowie - wie könnte es anders sein, in einer alten Schottischen Burg - böse Geister, die Malcolm nach dem Leben trachten. Die Geister werden vom Computer gesteuert und sind grün.

Das Spiel beginnt mit einem Geist als Gegner und endet mit dreien; die Zeit zum Bergen des Schatzes wird vom Spieler vorher festgelegt (Level 0 = 1000 Sekunden bis Level 9 = 10000 Sekunden). Malcolm muß alle Dukaten aufsammeln (sie zählen der aufgedruckten Wert, 1, 5, 10 Punkte) und die Geldsäcke (je 100 Punkte) aus dem hintersten Winkel der Burg holen. Danach muß er, um zu gewinnen, die Burg wieder durch eine der beiden Eingangstüren verlassen, durch die er sie betreten hat. Pro Runde sind 750 Punkte zu gewinnen. Sobald Malcolm seinen Auftrag einmal nicht erfüllen kann, wird die Punkteliste angezeigt.

Hier noch einige Erklärungen zum Spielablauf:

Neu bei diesem Spiel ist, daß der Spieler beim Aufbau der Grafik zusehen kann. Er sollte sich also schon ein wenig orientieren, bevor der Startsignalton zu hören ist. Dann sollte er die Burg so schnell wie möglich betreten, bevor sich die Eingangstore schließen. Wird Malcolm nämlich eingesperrt, bleibt ihm nichts anderes übrig, als darauf zu hoffen, daß sich die Türen irgendwann einmal wieder öffnen, bevor die Zeit um ist, oder die Geister ihn erreichen. Diese sind nämlich durch geschlossene Türen nicht aufzuhalten (Ausnahme: Die Eingangstüren der Burg)!

Vorsicht ist also geboten - vor allem beim Betreten von Sackgassen. Sobald man von den Geistern erwischt wird, ist das Spiel aus!

Während des Spiels wird der vorherangewählte Level, die erreichte Punktzahl und die noch zur Verfügung stehende Zeit angezeigt. Wenn Malcolm Punkte sammelt, ertönt ein Signal (je nach der Höhe der Punktzahl). Schafft er 750 Punkte und kann die Burg verlassen, so geht das Spiel, nach drücken der Enter-Taste, weiter.

Läuft die Zeit ab oder wird Malcolm von Geistern erwischt, so endet das Spiel und die Punkteliste wird ausgegeben. Sie zeigt an: Die absolute Höchstpunktzahl (750 a Runde); den Highscore mit Namen und Punktzahl, sowie das zuletzt erreichte Punktergebnis. Ist dieses Ergebnis höher als der Highscore, muß der Spielername (max 7 Großbuchstaben) eingegeben werden. Der Highscore und der Name wird dann automatisch geändert. Der Spieler kann jetzt wählen:

- 1. Weiterspielen (Highscore und Level werden übernommen).
- 2. Neues Spiel (Alle Variablen werden gelöscht, das Spiel beginnt von vorne).
- 3. Spielende, endgültiger Spielabbruch.

Die Eingabe der Wahlmöglichkeiten erfolgt über Call Key.

Besonderheit:

Das Programm kann mühelos erweitert werden, wenn die Memory Expansion angeschlossen ist. Das Spiel besitzt bewegte Grafik.

Programmaufteilung:

Zeilennummern:	An der Daten
2 - 30:	Programmkopf
32:	Aufruf Sub Vorspann ()
34 - 50:	Optionen und Images
52 - 72:	Verlauf und Weiterspielen nach gewonnener Runde
74 - 100:	Spielende/Weiterspielen bei verlorener Runde
102 - 104:	Programmende
106 - 136:	Hauptprogramm (Figurensteuerung)
138 - 206:	Spielerfigur-Bodenkontakte (öffnen und schließen der Türen, Punkte)
208 - 264:	Computerfiguren-Bodenkontakte
266 - 274:	Sprite Spielerfigur + Magnify
276 - 292:	Sprite Computerfigur
294 - 406:	Sub Grafik
ab 298:	- Umcodierung der ASCII-Zeichen
ab 320:	- Aufbau der Hauptgrafik
ab 364:	- Farbgebung
ab 382:	- Grafikdatei
408 - 448:	Sub Figuren
ab 412:	- Bilden der Dukatenfelder
ab 440:	- Bilden der Bodenkontaktfelder
450 - 458:	Sub Comp ()
	- Vergl. Spritegrafik + Zeichengrafik
460 - 526:	Sub Vorspann ()
ab 464:	- Titelbild-Ausgabe
ab 484:	- Spielregel
ab 522:	- Eingabe Zeitbegrenzung (Level)
526:	Subend (Ende des Programms)

```
2 !
4 ! ************************
6 ! *                      *
8 ! *  M U R C A L D Y     *
10 !*                       *
12 !*    C A S T L E        *
14 !*                       *
16 !*  ****************     *
18 !*                       *
20 !*     (C) 11/1983       *
22 !*                       *
24 !*  BY  H A G E R A      *
26 !*                       *
28 !************************
30 !
32 CALL VORSPANN(T)
34 !
36 !
38 ! OPTIONEN
40 !
42 IMAGE #######
44 IMAGE ######## ########## ########
46 RANDOMIZE :: ON WARNING NEXT :: ON BREAK NEXT
48 G=1
50 !
52 ! VORLAUF
54 CALL CLEAR
56 CALL GRAPHIK :: CALL FIGUREN
58 !
60 DISPLAY AT(22,1)SIZE(7):"LEVEL";T :: DISPLAY AT(23,1)SIZE(7):"PUNKTE:" :: DIS
PLAY AT(24,1)SIZE(7):USING 42:P
62 GOSUB 266 :: GOSUB 276 :: DISPLAY AT(22,23):"TIME:" :: TIME=10000-(T*1000)::
CALL SOUND(1000,400,0):: GOSUB 106
64 CALL DELSPRITE(ALL):: IF WIN=1 THEN WIN=0 ELSE GOTO 74
66 DISPLAY AT(22,21):"GAME" :: DISPLAY AT(23,21):"OVER!" :: DISPLAY AT(24,21):"Y
OU WIN!" :: DISPLAY AT(22,1)SIZE(7):"<ENTER>"
68 DISPLAY AT(23,1)SIZE(7):"LEVEL";T
70 CALL KEY(0,KW,KS):: IF KS=0 OR KW<>13 THEN 70
72 GOTO 52
74 ! VERLOREN
76 !
78 CALL DELSPRITE(ALL):: CALL CLEAR
80 DISPLAY AT(1,1):"MURCALDY CASTLE":"*****************************":"":"PUNKTEST
AND:"
82 DISPLAY AT(6,1):USING 44:"ABSOLUTE","PUNKTZAHL:","750 A.R."
84 DISPLAY AT(8,1):USING 44:"HISCORE",N$,HI
86 DISPLAY AT(10,1):USING 44:"LETZTES","ERGEBNIS :",P
88 IF P>HI THEN DISPLAY AT(12,1):"IHR NAME BITTE    :" :: ACCEPT AT(12,21)VALIDA
TE(UALPHA):N$ :: N$=N$&RPT$(" ",9-LEN(N$))&":" :: HI=P :: GOTO 84
90 P=0 :: G=1
92 !
94 DISPLAY AT(14,1):"*****************************":"":"BITTE WAEHLEN SIE:":"<1>
WEITERSPIELEN":"<2> NEUES SPIEL":"<3> SPIELENDE"
96 CALL KEY(0,KW,SW):: IF SW=0 OR(KW<49 OR KW>52)THEN 96
98 IF KW=49 THEN 52
100 IF KW=50 THEN RUN
102 END
104
106 ! HAUPTPROGRAMM
108 !
110 CALL MOTION(#1,0,0):: CALL JOYST(1,X,Y):: CALL POSITION(#1,XP,YP):: XC1=INT(
(XP+6)/8)+1 :: YC1=INT((YP+6)/8)+1 :: CALL GCHAR(XC1-Y/4,YC1+X/4,CG1)
112 XC2=INT((XP+7)/8)+1 :: YC2=INT((YP+7)/8)+1 :: CALL GCHAR(XC2-Y/4,YC2+X/4,CG2
)
114 IF CG1>99 AND CG2>99 THEN CALL MOTION(#1,-Y/4,X/4)
116 CALL COMP(1,FX,FY,CC):: ON INT((CC-86)/10)GOSUB 142,144,146,148,150
118 IF P=750*(G-1)AND CC=101 THEN WIN=1 :: RETURN
120 FOR N=2 TO G :: CALL MOTION(#N,0,0)
```

Murcaldy Castle

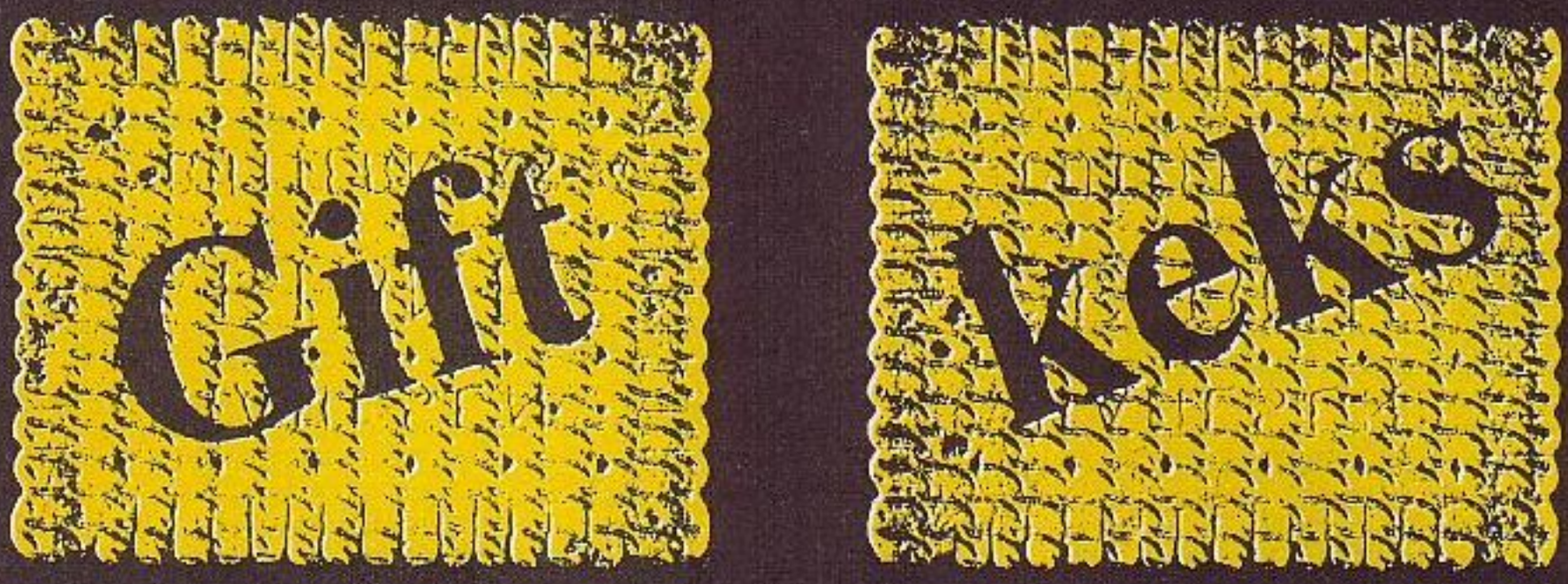
```
122 CALL POSITION(#N,XPG(N),YPG(N)):: IF XP<XPG(N)AND YP<YPG(N)THEN IF RND*2<1 T
HEN X=-4 :: Y=0 ELSE Y=-4 :: X=0
124 IF XP<XPG(N)AND YP>YPG(N)THEN IF RND*2<1 THEN X=-4 :: Y=0 ELSE Y=4 :: X=0
126 IF XP>XPG(N)AND YP<YPG(N)THEN IF RND*2<1 THEN X=4 :: Y=0 ELSE Y=-4 :: X=0
128 IF XP>XPG(N)AND YP>YPG(N)THEN IF RND*2<1 THEN X=4 :: Y=0 ELSE Y=4 :: X=0
130 XC1=INT((XPG(N)+6)/8)+1 :: YC1=INT((YPG(N)+6)/8)+1 :: CALL GCHAR(XC1+X/4,YC1
+Y/4,CGG1)
132 XC2=INT((XPG(N)+7)/8)+1 :: YC2=INT((YPG(N)+7)/8)+1 :: CALL GCHAR(XC2+X/4,YC2
+Y/4,CGG2):: IF CGG1>97 AND CGG2>97 THEN CALL MOTION(#N,INT(X/4),INT(Y/4))
134 CALL COMP(N,FX,FY,CC):: ON INT((CC-86)/10)GOSUB 212,214,216,218,220 :: CALL
COINC(#1,#N,8,COI):: IF COI=-1 THEN RETURN
136 NEXT N :: DISPLAY AT(24,23)SIZE(6):TIME :: TIME=TIME-1 :: IF TIME=0 THEN RET
URN ELSE 110
138 ! MANN-FELDER
140 !
142 ON CC-95 GOTO 154,154,154,154,154,154,154,154,156,158
144 ON CC-105 GOTO 160,162,164,166,168,170,172,174,176,178
146 ON CC-115 GOTO 180,182,184,186,188,154,154,154,154,154
148 ON CC-125 GOTO 154,154,196,154,154,154,154,154,154,154
150 ON CC-135 GOTO 204,154,154
152 !
154 RETURN
156 CALL VCHAR(2,13,99,2):: CALL HCHAR(10,25,102,2):: GOTO 154
158 CALL VCHAR(2,20,98,2):: CALL HCHAR(18,16,102,2):: GOTO 154
160 CALL HCHAR(4,2,99,2):: CALL HCHAR(18,7,102,2):: GOTO 154
162 CALL HCHAR(4,30,99,2):: CALL VCHAR(11,29,102,2):: GOTO 154
164 CALL HCHAR(7,2,99,2):: CALL VCHAR(10,19,102,2):: GOTO 154
166 CALL HCHAR(7,8,99,2):: CALL VCHAR(10,14,102,2):: GOTO 154
168 CALL HCHAR(7,24,99,2):: CALL VCHAR(11,4,102,2):: GOTO 154
170 CALL HCHAR(7,30,99,2):: CALL VCHAR(2,13,102,2):: GOTO 154
172 CALL MOTION(#1,0,0):: P=P+1 :: CALL HCHAR(FX,FY,103):: CALL SOUND(-200,220,0
):: DISPLAY AT(24,1)SIZE(7):USING 42:P :: GOTO 154
174 CALL VCHAR(11,4,98,2):: CALL VCHAR(2,20,102,2):: GOTO 154
176 CALL VCHAR(10,14,98,2):: CALL HCHAR(4,30,102,2):: GOTO 154
178 CALL VCHAR(10,19,98,2):: CALL HCHAR(4,2,102,2):: GOTO 154
180 CALL VCHAR(11,29,98,2):: CALL HCHAR(7,2,102,2):: GOTO 154
182 CALL HCHAR(18,7,99,2):: CALL HCHAR(7,8,102,2):: GOTO 154
184 CALL HCHAR(18,16,99,2):: CALL HCHAR(7,24,102,2):: GOTO 154
186 CALL HCHAR(18,25,99,2):: CALL HCHAR(7,30,102,2):: GOTO 154
188 CALL MOTION(#1,0,0):: P=P+5 :: FOR F=440 TO 220 STEP -55 :: CALL SOUND(-200,
F,0):: NEXT F
190 CALL HCHAR(FX,FY,103):: DISPLAY AT(24,1)SIZE(7):USING 42:P :: GOTO 154
192 !
194 !
196 CALL MOTION(#1,0,0):: P=P+10 :: FOR F=400 TO 260 STEP -20 :: CALL SOUND(-200
,F,0):: NEXT F
198 CALL HCHAR(FX,FY,103):: DISPLAY AT(24,1)SIZE(7):USING 42:P :: GOTO 154
200 !
202 !
204 CALL MOTION(#1,0,0):: P=P+100 :: FOR F=250 TO 400 STEP 10 :: CALL SOUND(-200
,F,0):: NEXT F
206 CALL HCHAR(FX,FY,103):: DISPLAY AT(24,1)SIZE(7):USING 42:P :: GOTO 154
208 ! GEIST-FELDER
210 !
212 ON CC-95 GOTO 224,224,224,224,224,224,224,224,226,228
214 ON CC-105 GOTO 230,232,234,236,238,240,224,242,244,246
216 ON CC-115 GOTO 248,250,252,254,256,258,260,262,264,32767
218 !
220 !
222 !
224 RETURN
226 CALL VCHAR(2,13,98,2):: CALL VCHAR(2,20,98,2):: GOTO 224
228 CALL HCHAR(4,2,99,2):: CALL HCHAR(4,30,99,2):: GOTO 224
230 CALL HCHAR(7,2,99,2):: CALL HCHAR(7,30,99,2):: GOTO 224
232 CALL HCHAR(7,8,99,2):: CALL HCHAR(7,24,99,2):: GOTO 224
234 CALL VCHAR(10,14,98,2):: CALL VCHAR(10,19,98,2):: GOTO 224
236 CALL VCHAR(11,4,98,2):: CALL VCHAR(11,29,98,2):: GOTO 224
238 CALL HCHAR(18,7,99,2):: CALL HCHAR(18,25,99,2):: GOTO 224
240 CALL HCHAR(18,16,99,2):: CALL HCHAR(21,12,97,2):: CALL HCHAR(21,20,97,2):: G
OTO 224
242 CALL VCHAR(2,13,102,2):: CALL VCHAR(2,20,102,2):: CALL HCHAR(18,16,102,2)::
GOTO 224
244 CALL HCHAR(7,2,102,2):: CALL HCHAR(18,7,102,2):: CALL VCHAR(10,14,102,2):: G
OTO 224
246 CALL HCHAR(4,30,102,2):: CALL HCHAR(18,25,102,2):: CALL VCHAR(10,19,102,2)::
 GOTO 224
248 CALL HCHAR(7,2,102,2):: CALL HCHAR(7,8,102,2):: CALL VCHAR(11,4,102,2):: GOT
O 224
250 CALL HCHAR(7,24,102,2):: CALL HCHAR(7,30,102,2):: CALL VCHAR(11,29,102,2)::
GOTO 224
252 CALL HCHAR(21,12,102,2):: CALL HCHAR(21,20,102,2):: GOTO 224
254 !
256 ! RESERVIERT FUER NACH-        TRAEGLICH EINSETZBARE        FUNKTIONEN, Z.B. F
ALL        TUEREN ETC.
```

```
258 !
260 !
262 !
264 GOTO 224
266 ! SPRITE MAN
268 !
270 CALL MAGNIFY(3)
272 CALL SPRITE(#1,132,4,169,121)
274 RETURN
276 ! SPRITE GHOST
278 !
280 G=G+1 :: IF G>4 THEN G=4
282 CALL SPRITE(#2,140,13,33,9):: IF G<3 THEN RETURN
284 CALL SPRITE(#3,140,13,35,233):: IF G<4 THEN RETURN
286 CALL SPRITE(#4,140,13,49,120)
288 !
290 !
292 RETURN
294 SUB GRAPHIC
296 !
298 CALL CHAR(96,"FFFFFFFFFFFFFFFFF000000FFFF000000181818181818E18000000FFFF0000
00")
300 CALL CHAR(101,"0000000000000000000000000000000000000000000000000000")
302 CALL CHAR(104,"C0000000000000000000000C0000000000000000000000000000000000000
000")
304 CALL CHAR(112,"3C66D7B7F7F7763C0000000000000000000000000000000000000000000
000")
306 CALL CHAR(108,"C0000000000000000000000000000000000000000000000000000000000
000")
308 CALL CHAR(120,"7E425E467A7A467E")
310 CALL CHAR(116,"0000000000000000000C0000000000000000000000000000000000000000
000")
312 CALL CHAR(128,"00FFC9A9E9E9FF00")
314 CALL CHAR(132,"00071C3046464041595C4F473010070000E033806262028829A3AF2E20C33E
000")
316 CALL CHAR(136,"00187E3C3C7E7E00")
318 CALL CHAR(140,"0001070F19191F1E1F18131F191919000080E0F09898F878F818C8F898787
800")
320 CALL HCHAR(1,1,103,768):: CALL HCHAR(22,12,101,10):: CALL HCHAR(23,12,101,10
)
322 RESTORE 384
324 READ R,C,W
326 IF R=0 THEN 332
328 CALL HCHAR(R,C,96,W)
330 GOTO 324
332 !
334 READ R,C,W
336 IF R=0 THEN 342
338 CALL VCHAR(R,C,96,W)
340 GOTO 334
342 !
344 READ R,C
346 IF R=0 THEN 352
348 CALL HCHAR(R,C,102,2)
350 GOTO 344
352 !
354 READ R,C
356 IF R=0 THEN 362
358 CALL VCHAR(R,C,102,2)
360 GOTO 354
362 !
364 CALL SCREEN(2)
365 CALL COLOR(9,7,15)
368 CALL COLOR(11,5,15)
370 CALL COLOR(12,13,15)
372 CALL COLOR(13,14,15)
374 FOR I=0 TO 8 :: CALL COLOR(I,5,15):: NEXT I
376 CALL COLOR(14,11,2)
378 CALL COLOR(10,15,15)
380 GOTO 406
382 ! CATEI
384 !
386 DATA 1,1,32,1,1,10,1,20,10,7,1,1,7,24,1,7,10,5,7,17,5,12,10,5,12,17,5,13,1,3
,13,27,3
388 DATA 15,8,2,6,13,8
390 DATA 15,15,1,15,24,2,10,1,3,18,9,7,18,18,7,13,27,3,21,2,10,21,14,6,21,22,10,
24,11,12,0,0,0
392 !
394 DATA 2,1,23,22,2,3,5,13,1,5,20,1,7,10,2,7,23,2,8,1,3,8,29,3,10,7,6,10,26,6,1
3,12,3,13,21,3
396 DATA 16,1,2,16,29,2,22,11,2,22,22,2,1,32,23,0,0,0
398 !
400 DATA 1,2,1,30,7,2,7,8,7,24,7,30,18,7,18,16,18,25,0,0
402 !
```

```
404 DATA 2,13,2,20,10,14,10,19,11,4,11,29,0,0
406 SUBEND
408 SUB FIGUREN
410 !
412 CALL HCHAR(5,14,136,6)
414 !
416 FOR Q=1 TO 25
418 I=INT(RND*16)+5 :: J=INT(RND*30)+2 :: CALL GCHAR(I,J,CG):: IF CG=103 THEN CA
LL HCHAR(I,J,112)ELSE 418
420 NEXT Q
422 !
424 FOR Q=1 TO 15
426 I=INT(RND*16)+5 :: J=INT(RND*30)+2 :: CALL GCHAR(I,J,CG):: IF CG=103 THEN CA
LL HCHAR(I,J,120)ELSE 426
428 NEXT Q
430 !
432 FOR Q=1 TO 5
434 I=INT(RND*16)+5 :: J=INT(RND*30)+2 :: CALL GCHAR(I,J,CG):: IF CG=103 THEN CA
LL HCHAR(I,J,120)ELSE 434
436 NEXT Q
438 !
440 FOR Q=1 TO 2 :: FOR CH=104 TO 119 :: IF CH=112 THEN 444
442 I=INT(RND*19)+2 :: J=INT(RND*30)+2 :: CALL GCHAR(I,J,CG):: IF CG=103 THEN CA
LL HCHAR(I,J,CH)ELSE 442
444 NEXT CH :: NEXT Q
446 !
448 SUBEND
450 SUB COMP(GH,FX,FY,CC)
452 !
454 CALL POSITION(#GH,DX,DY):: FX=INT((DX+7)/8)+1 :: FY=INT((DY+7)/8)+1 :: CALL
GCHAR(FX,FY,CG):: CC=CG
456 !
458 SUBEND
460 SUB VORSPANN(T)
462 !
464 CALL CLEAR
466 CALL SCREEN(15):: FOR I=0 TO 8 :: CALL COLOR(I,5,15):: NEXT I
468 DISPLAY AT(1,1):"******************************" :: DISPLAY AT(24,1):"********
**********************"
470 DISPLAY AT(3,1):"    H A G E R A     ":"":" G R A P H I K - G A M E S
 ":"":"":"M U R C A L D Y  C A S T L E"
472 DISPLAY AT(12,1):"      ****   ****   ****          ****   ****   ****
     ****  ****  ****            ***  ****  ***"
474 DISPLAY AT(16,1):"         ************              ************
       ************"
476 DISPLAY AT(20,1):"BITTE WAEHLEN SIE:":"":"<1> SPIELSTART":"<2> SPIELREGEL"
478 CALL KEY(0,KW,SW):: IF SW=0 OR(KW<>49 AND KW<>50)THEN 478
480 IF KW=49 THEN 522
482 !
484 CALL CLEAR
486 DISPLAY AT(1,1):"MURCALDY CASTLE":"MALCOLM MCMONEY SOLL EINEN  SCHATZ AUS DE
N ALTEHRWUERDI=GEN GEMAEUERN VON MURCALDY".
488 DISPLAY AT(5,1):"CASTLE BERGEN. DIE AUFGABE  IST NICHT EINFACH, DENN UE= BER
ALL GIBT ES VERSTECKTE  BODENKONTAKTE, WELCHE DIE"
490 DISPLAY AT(9,1):"TUEREN INNERHALB DER BURG  OEFFNEN UND SCHLIESSEN. AUS=SER
DEM WIRD MURCALDY CASTLE VON BOESEN GEISTERN BEWACHT!"
492 DISPLAY AT(13,1):"VON DIESEN DARF SICH MALCOLMMCMONEY NICHT ERWISCHEN LAS=SE
N."
494 DISPLAY AT(17,1):"MALCOLM MCMONEY BETRITT MUR=CALDY CASTLE DURCH EINE DER BE
IDEN PFORTEN UND MUSS ALLE(NZELNEN DUKATEN SOWIE DIE"
496 DISPLAY AT(21,1):"GOLDSAECKE IM HINTEREN RAUM EINSAMMELN":"":"<ENTER>"
498 CALL KEY(0,KW,SW):: IF SW=0 OR KW<>13 THEN 498
500 CALL CLEAR
502 DISPLAY AT(1,1):"DANACH MUSS ER DIE BURG WIE=DER DURCH EINES DER BEIDEN  TOR
E VERLASSEN. WENN ER SEI=NEN AUFTRAG ERFUELLEN KANN,"
504 DISPLAY AT(5,1):"WARTET SOGLEICH EIN NEUER  AUF IHN: ER MUSS DEN NAECH= STE
N SCHATZ BERGEN - ALLER= DINGS SIND DIE GEISTER NUN"
506 DISPLAY AT(9,1):"WACHSAMER UND IN DER MEHR=  ZAHL! WIEVIEL ZEIT MALCOLM  MCM
ONEY FUER SEINEN AUFTRAG HAT, LEGEN SIE SELBST FEST!"
508 DISPLAY AT(14,1):"PUNKTESCHEMA:":"DIE DUKATEN ZAEHLEN DEN AUF=GEDRUCKTEN WER
T (1,5,10 P.).":"DIE GOLDSAECKE ZAEHLEN JE  100 PUNKTE."
510 DISPLAY AT(19,1):"WENN MALCOLM MCMONEY LANGE  EINGESCHLOSSEN IST ODER VON GE
ISTERN ERWISCHT WIRD, KANNER SEINEN AUFTRAG NICHT ER="
512 DISPLAY AT(23,1):"FUELLEN. ER STIRBT!":"<ENTER>"
514 CALL KEY(0,KW,SW):: IF SW=0 OR KW<>13 THEN 514
516 CALL CLEAR
518 DISPLAY AT(1,1):"WARNUNG!":"DIE GEISTER KOEN=NEN GESCHLOSSENE TUEREN MIT AUS
NAHME DER HAUPTEINGAENGE DURCHDRINGEN. MCMONEY MUSS"
520 DISPLAY AT(6,1):"ALSO BESONDERS VORSICHTIG  SEIN, WENN ER SICH IN SACK= GAS
SEN HINEINWAGT!"
522 DISPLAY AT(20,1):"WIE LANGE SOLL MCMONEY ZEIT HABEN, DEN SCHATZ ZU BERGEN?":
"(LEVEL 0-9):" :: ACCEPT AT(22,15)SIZE(1)VALIDATE(DIGIT):T
524 CALL CLEAR
526 SUBEND
```

TI-99/4A

Man könnte meinen, daß das Spiel "Giftkeks" für den TI-99, aus einem Agatha-Christie-Krimi entnommen wurde. Denn genau wie die bekannte englische Kriminalschriftstellerin, die es vorzüglich verstand, ihre ausgewählten Opfer auf immer wieder neue und heimtückische Art und Weise in das Jenseits zu befördern (natürlich nur auf dem Papier), hat auch der Ersteller dieses Programms an den Nervenkitzel der Teilnehmer gedacht.



Spielablauf:
Sie rufen, am besten an einem nebligen, stürmischen und gewittrigen Abend (muß zwar nicht sein, würde allerdings zur passenden Atmosphäre eines Agatha Christie Krimis beitragen) Freunde und Verwandte (bis 9 Personen) zusammen - wobei die letztere Gruppe einer gründlichen Gewissensprüfung zu unterziehen ist, da man Ihnen die Einladung leicht übelnehmen könnte.
Nachdem sich alle vor dem Giftmörder, sprich Computer, plaziert haben, kann es los gehen!
Ist das Laden der Kassette beendet, erscheint auf dem Bildschirm Ihres TI-99 ein wirklich harmlos aussehender, sehr appetitlicher Keks, der geradezu auffordert, in ihn hineinzubeißen.
Doch größte Vorsicht ist geboten:
Der Keks ist präpariert, wahrscheinlich mit einem Gift, das kein Arzt der Welt in Ihrem Körper ausfindig machen kann (wir nennen hier absichtlich keinen Namen).
Nun darf jeder Teilnehmer in den Keks beißen. Dies macht er nicht wahllos, sondern er kombiniert seinen Biß so, damit das Stück mit dem Gift, für den nächsten Spieler übrig bleibt.
Na dann viel Vergnügen!
P. S.: Am Anfang des Spiels kann man die Größe des Kekses beliebig festlegen. Je größer der Keks, desto schwieriger das Spiel. Demonstrierend ist hier auch die gute Grafik des TI-99.

Giftkeks

```
100 REM **********************
110 REM * COPYRIGHT BY      *
120 REM * HEIKO BECKE       *
130 REM * 4750 UNNA         *
140 REM * WIESENSTRASSE 3   *
150 REM **********************
160 N$="07081212100804Ø5E01048480810208Ø043338070330300020CC1CE0C03C0C00" :: CAL
L CHAR(104,N$,64,"3C4299A1A199423C"):: K=11 :: K1=1 :: M$=RPT$("F",32)
170 CALL CLEAR :: CALL COLOR(10,16,7):: CALL COLOR(9,11,1):: CALL SCREEN(2):: FO
R F1=1 TO 8 :: CALL COLOR(F1,14,1):: NEXT F1
180 CALL COLOR(0,15,1)
190 DISPLAY AT(1,11)ERASE ALL:"GIFTKEKS"
200 DIM A(10,10)
210 DISPLAY AT(12,1):"SPIELREGELN? (J)A, (N)EIN":  :  :  :  :  :  :  :  :TAB(8);"@ HEI
KO BECKE 3.1983"
220 CALL KEY(0,KEY,STATUS):: IF STATUS=0 THEN 220 ELSE IF KEY=78 THEN 470 ELSE I
F KEY<>74 THEN 220
230 F=1 :: R=7 :: C=5 :: DISPLAY AT(1,1)ERASE ALL:"GIFTKEKS IST FUER 1 ODER":  :"
MEHRERE SPIELER."
```

```
240 DISPLAY AT(22,1):"(DIESES IST VON DER GROESSE"; :TAB(20);"5 MAL 7)" :: GOSUB
 530 :: FOR Q=1 TO 1000 :: NEXT Q
250 DISPLAY AT(1,1)ERASE ALL:"DER GIFTKEKS IST EIN":"RIESIGES PLAETZCHEN":"Z REI
HEN HOCH UND":"C SPALTEN BREIT."
260 DISPLAY AT(6,1):"DU GIBST AM ANFANG Z UND S":"EIN. AN DER OBEREN LINKEN":"EC
KE DES PLAETZCHENS IST EIN":"VERGIFTETES QUADRAT."
270 DISPLAY AT(11,1):"WER DAS GIFTQUADRAT ABBEISST":"VERLIERT. ZUM BEISSEN GEBE"
:"REIHE UND SPALTE EINES QUA-":"DRATES DES PLAETZCHENS EIN."
280 DISPLAY AT(16,1):"ALLE QUADRATE UNTERHALB UND" "RECHTS DIESES QUADRATES":"(E
INSCHLIESSLICH DIESES QUA-":"DRATES) VERSCHWINDEN."
290 DISPLAY AT(20,1):"-------KAU----SCHLUCK-------":"DAS ABBEISSEN SCHON GEGESSE
-":"NER QUADRATE SOWIE JENER":"AUSSERHALB DER ECHTEN DIMEN-"
300 DISPLAY AT(24,1):"SIONEN IST UNGUELTIG.(ENTER)" :: CALL KEY(0,KEY,STATUS)::
IF STATUS=0 OR KEY<>13 THEN 300
310 GR=GR+1 :: IF GR>13 AND K1=8 THEN GR=1 ::K1=10 :: K=2
320 IF GR>13 THEN GR=1 :: K1=8 :: K=2
330 ON GR GOTO 340,350,360,370,380,390,400,410,420,430,440,450,460
340 M$=RPT$("AA55",8):: GOTO 470
350 M$=RPT$("55AAAA",6):: GOTO 470
360 M$=RPT$("55AA55",6):: GOTO 470
370 M$=RPT$("3966CCCCCCCC6633",2):: GOTO 470
380 M$=RPT$("AD4295A8299542B5",2):: GOTO 470
390 M$=RPT$("CCCC3333",4):: GOTO 470
400 M$=RPT$("18244299994224l8",2):: GOTO 470
410 M$=RPT$("384482AAAA824438",2):: GOTO 470
420 M$="3449423108834498C6292511A221120C" :: GOTO 470
430 M$=RPT$("669981814242241B",2):: K=9 :: GOTO 470 !HERZ
440 M$="3C42AA81AA99423C06C90AC925D926:8" :: K=4 :: K1=1 :: GOTO 470 !WURM
450 M$="C2310C43A30C31C2431C30C5C2301C23" :: K=16 :: GOTO 470
460 M$=RPT$("FF99A5C3C3A599FF",2):: K=1 :: K1=6
470 DISPLAY AT(3,8)ERASE ALL:"FANGEN WIR AN..." :: F=0 :: FOR I=1 TO 10 :: FOR J
=1 TO 10 :: A(I,J)=0 :: NEXT J :: NEXT I
480 ON BREAK STOP :: A$="WIEVIELE" :: DISPLAY AT(6,1):F$;" SPIELER?": :A$;" SPAL
TEN?": :A$;" ZEILEN?"
490 ON WARNING NEXT
500 ACCEPT AT(6,19)SIZE(-1)BEEP VALIDATE("123456789"):F :: I1=0 :: ACCEPT AT(8,1
9)SIZE(-1)BEEP VALIDATE("123456789"):C
510 ACCEPT AT(10,19)SIZE(-1)BEEP VALIDATE("123456789"):R
520 C2=C :: R2=R :: CALL CLEAR
530 FOR I=1 TO C :: FOR J=1 TO R :: A(I,J)=1 :: NEXT J :: NEXT I :: A(1,1)=-1
540 CALL CHAR(96,M$):: CALL COLOR(9,K,K1)
550 ! DRUCKE DAS SPIELBRETT
560 ON BREAK NEXT :: Q=5 :: FOR I=1 TO R :: Q=Q+2 :: DISPLAY AT(Q,6)SIZE(2):I ::
 NEXT I
570 T=8 :: FOR I=1 TO C :: T=T+2 :: DISPLAY AT(5,T)SIZE(2):I :: NEXT I
580 ! PLAETZCHEN
590 FOR SPALTE=C TO 1 STEP -1 :: T=2*SPALTE+9 :: FOR ZEILE=R TO 1 STEP -1 :: Q=2
*ZEILE+5
600 IF A(SPALTE,ZEILE)=-1 THEN 620 ELSE IF A(SPALTE,ZEILE)=0 THEN 630
610 DISPLAY AT(Q+1,T)SIZE(2):CHR$(96)&CHR$(97):: DISPLAY AT(Q,T)SIZE(2):CHR$(96)
&CHR$(97):: GOTO 640
620 DISPLAY AT(Q+1,T)SIZE(2):CHR$(106)&CHR$(107):: DISPLAY AT(Q,T)SIZE(2):CHR$(1
04)&CHR$(105):: GOTO 640
630 DISPLAY AT(Q+1,T)SIZE(-2):"  " :: DISPLAY AT(Q,T)SIZE(-2):"  "
640 NEXT ZEILE
650 NEXT SPALTE
660 IF F=0 THEN 670 ELSE RETURN
670 ! ERHALTE ABWECHSLUNGSWEISE DIE BISSE DER SPIELER
680 I1=I1+1 :: P1=I1-INT(I1/P)*P :: IF P1<>0 THEN 690 ELSE P1=P
690 DISPLAY AT(1,1):"SPIELER ";P1: :"KOORDINATEN DES BISSES:"
700 ON WARNING NEXT
710 ACCEPT AT(3,25)SIZE(-3)BEEP VALIDATE(DIGIT,","):F$ :: IF LEN(F$)<>3 OR SEG$(
F$,1,1)="," OR SEG$(F$,3,1)="," OR SEG$(F$,2,1)<>"," THEN 710
720 C1=VAL(SEG$(F$,1,1)):: R1=VAL(SEG$(F$,3,1))
730 IF R1<1 OR R1>R OR C1<1 OR C1>C OR A(C1,R1)=0 THEN 740 ELSE IF A(C1,R1)=-1 T
HEN 770 ELSE 750
740 DISPLAY AT(1,1):"UNGUELTIG. DU VERSUCHST EIN": :"LEERES QUADRAT ABZUBEISSEN!
" :: FOR I=1 TO 450 :: NEXT I :: GOTO 690
750 FOR I=C1 TO C :: FOR J=R1 TO R :: A(I,J)=0 :: NEXT J :: NEXT I :: GOTO 550
760 ! ENDE DES SPIELES IN ZEILE 730 ERKANNT
770 FOR V1=1 TO 15 :: FOR V=1 TO 5 :: CALL COLOR(10,7,16):: NEXT V :: CALL COLOR
(10,16,7):: NEXT V1
780 DISPLAY AT(1,1):"SPIELER ";P1;", DU VERLIERST." :: CALL HCHAR(3,3,32,28)
790 ! SOUND
800 FOR I=1 TO 10 :: READ Z(I),S(I):: CALL SOUND(Z(I),S(I),9):: NEXT I :: RESTOR
E
810 DATA 200,198,200,190,700,198,200,149,200,157,200,150,200,140,500,123,200,133
,700,133
820 DISPLAY AT(5,1)SIZE(4):"NOCH" :: DISPLAY AT(7,1)SIZE(3):"EIN" :: DISPLAY AT(
9,1)SIZE(6):"SPIEL?"
830 DISPLAY AT(13,1)SIZE(4):"(J)A" :: DISPLAY AT(15,1)SIZE(6):"(N)EIN"
840 CALL KEY(0,KEY,STATUS):: IF STATUS=0 THEN 840 ELSE IF KEY=74 THEN 310 ELSE I
F KEY<>78 THEN CALL SOUND(4250,-4,0):: GOTO 840 ELSE CALL CLEAR :: END
```

Dynamics Arbeitsprogramm "Lagerkartei" – für TI-99/4A

Über kurz oder lang entscheiden sich viele Betriebe, in letzter Zeit besonders Kleinbetriebe, zur Rationalisierung, Systematisierung und Katalogisierung von Waren, Computer einzusetzen. Das Programm "Lagerkartei" stellt eine optimale Lösung dar, diese Vorhaben mit Hilfe eines TI-Computers zu realisieren.
"LAGERKARTEI" ist in Extended BASIC geschrieben, menuegesteuert und sehr benatzerfreundlich. Es erlaubt die Eingabe von maximal 100 Artikeln. Zu jedem Artikel können Artikelname, Artikelnummer, Preis und ein Zusatz eingegeben werden. Jede Eingabe ist auf maximal 16 Zeichen begrenzt.
Der Anwender hat die Möglichkeit, Artikel einzugeben, zu lesen, zu ändern, abzuspeichern, einzulesen, zu drucken und zu ordnen. Zum Lesen eines Artikels kann ein beliebiges Feldelement, z. B. eine Artikelnummer oder der Preis, eingegeben werden. Es besteht weiterhin die Möglichkeit, einzelne Artikel, alle Artikel oder Artikel nach bestimmten Kriterien ausdrucken zu lassen. Durch blockweises Abspeichern der Datensätze läßt sich eine Lagerkartei aufbauen, die mehrere hundert Artikel verwaltet.

Dynamics Lernprogramm "City-Quiz" – für TI-99/4A

"CITY-QUIZ" ist ein Übungs- und Lernprogramm. Es ist so konzipiert, daß vorhandenes Wissen vertieft wird, bzw. neue Kenntnisse vermittelt werden. Der Computer wählt aus 28 europäischen Ländern ein Land aus, dessen Hauptstadt zu ermitteln ist. Um den richtigen Namen der Hauptstadt zu erraten, sind fünf Versuche möglich. Nach jeder falschen Antwort wird auf dem Bildschirm ein Buchstabe des Namens der Stadt angezeigt. Die Buchstaben werden vom Computer zufällig ausgewählt. Nach der fünften falschen Antwort wird der richtige Name genannt. Man kann sich dann entscheiden, ob man das Programm beenden will oder weitere Hauptstädte erraten möchte.

Dynamics Lernprogramm "Vokabel-Trainer" für TI-99/4A

Der "Vokabel-Trainer" ist ein universell verwendbares Lern- und Übungsprogramm, das sich für jede Fremdsprache verwenden läßt. Diese Programmist ein gutes Beispiel für das programmierte Lernen von Vokabeln. Es hat unter anderem den Vorteil, daß der Anwender sein Lerntempo selbst bestimmen kann.
Es können maximal 200 Vokabeln (100 Fremdsprachenvokabeln und 100 deutsche Vokabeln) eingegeben werden. Die eingegebenen Vokabeln können entweder auf Diskette oder Kassette abgespeichert werden. Der Benutzer erhält somit die Möglichkeit, eine Vokabel-Bibliothek anzulegen.
Die Vokabeln können blockweise gelernt werden. Der Anwender hat die Wahl, Deutsch-Fremdsprache oder Fremdsprache zu üben. Nach Eingabe einer Vokabel wird die Richtigkeit der Antwort überprüft. Bei einer falschen Antwort wird die richtige Vokabel angezeigt. Wird das Programm beendet, so erscheinen auf dem Bildschirm die Summen der geübten, der richtig eingegebenen und der falsch geschriebenen Vokabeln. Der Anwender kann somit seinen Lernzuwachs überprüfen.

Dynamics Arbeitsprogramm "Data-Bank" – für TI-99/4A

"DATA-BANK" ist ein sehr komfortables und universell einsetzbares Dateiverwaltungsprogramm. Es läßt sich z. B. für die Adressenverwaltung oder Mitgliederverwaltung einsetzen. Maximal können 45 Datensätze eingegeben werden. Jeder Datensatz kann bis zu 120 Zeichen enthalten. Jeder Datensatz umfaßt 8 Felder, wobei jedes Feld auf 16 Zeichen begrenzt ist.
Das Programm erlaubt die Eingabe, das Lesen, Ändern, Abspeichern, Einlesen, Drucken und Ordnen von Daten. Es besteht die Möglichkeit, nach allen Feldern zu ordnen und zu selektieren. Zum Lesen eines Datensatzes kann ein beliebiges kriterium, z. B. eine Telefonnummer oder Straße, eingegeben werden. Ist dieses Kriterium in irgendeinem Datensatz vorhanden, so wird dieser angezeigt. Der Benutzer hat die Möglichkeit, weitere

Datensätze zu lesen, drukken zu lassen oder nach weiteren zu suchen, die das gleiche Kriterium enthalten. Datensätze können entweder nach Datensatznummern oder nach bestimmten Kriterien ausgedruckt werden.
"DATA-Bank" ist in Extenden BASIC geschrieben, menuegesteuert und sehr benutzerfreundlich. Es läßt sich daher sowohl im kommerziellen als auch im häuslichen Bereich verwenden.

Dynamics Zubehör

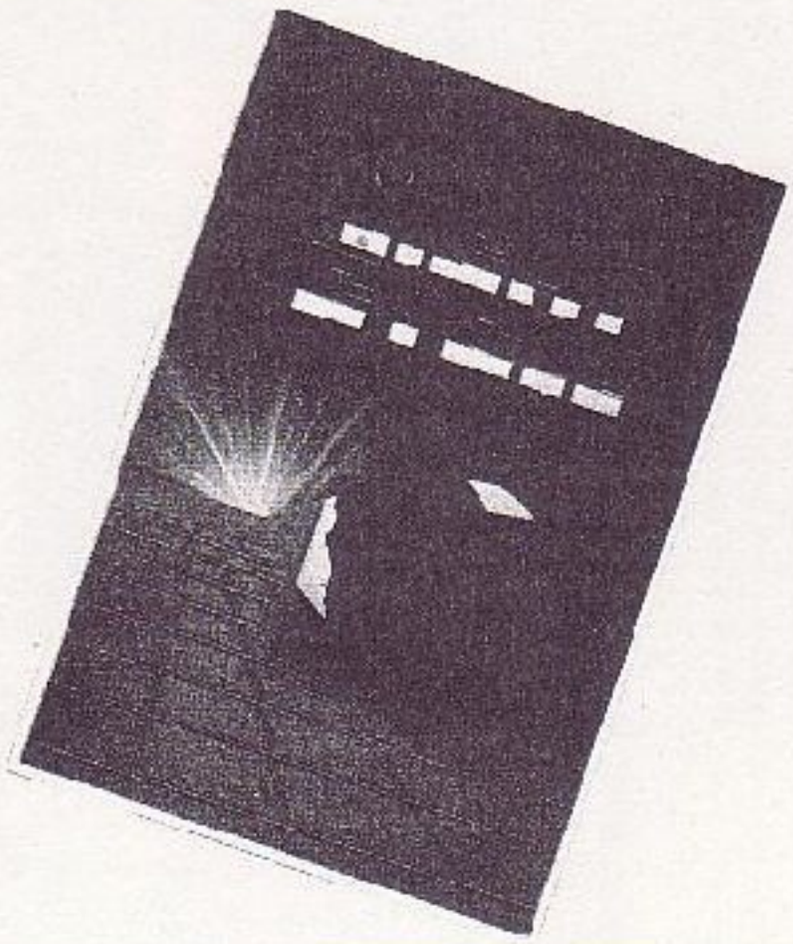
Joystick-Adapter für Coleco-Vision
Dieses Y-Adapterkabel ermöglicht es, alle Joysticks und andere Bedienungs-Elemente passend für Atari an der gleichen Eingangsbuchse und zur gleichen Zeit, mit dem original KEY-PAD Bedienungs-Element am Coleco-Vision-Gerät anzuschließen und zu bedienen.

Joystick-Adapter für Texas Instruments
Dieses Y-Adapterkabel ermöglicht es, alle Joysticks und Bedienungs-Elemente passend für Atari an das Texas Instruments-Gerät TI-99/4A anzuschließen. Eine eingebaute Schaltung schaltet automatisch zwischen 1. und 2. Spieler.

Verlängerungskabel für Joysticks
Für mehr Bewegungsfreiheit am Computer sorgt das Verlängerungskabel. 1 Paar mit jeweils 2 Meter Länge, für die linken und rechten Bedienungs-Elemente wie Joysticks, Track-Balls, Lichtzeichner usw.

Schnell-Feuer-Adapter für Joysticks
Dieser Adapter macht aus der Feuertaste jedes Joysticks eine Schnellfeuerwaffe. Die Schußgeschwindigkeit ist am Adapter von 1 bis 20 Schuß pro Sekunde einstellbar. Der Anschluß ist einfach und bedarf keiner technischen Eingriffe am Basisgerät oder Joystick.

Light Cycle für den Oric-1



Der Sinn dieses Spieles ist es, mit dem Laser entlang des Netzwerkes zu fahren und sich zu seiner Verteidigung einen soliden Schutzschirm aufzubauen. Der Gegenspieler, entweder Mensch oder Computer, versucht das gleiche. Allen im Wege liegenden Objekten müssen Sie ausweichen, sonst gehen Sie in einem spektakulären Lichtblitz unter. Das Programm ist sehr gut gestaltet und verlangt höchste Geschicklichkeit von seinem Benutzer. Eine Menge von Geräuscheffekten und hübschen Grafikspielchen ist eingebaut. Die Anleitung ist klar und übersichtlich. Die Bedienungstasten (hoch, herunter, rechts, links und Feuer) sind sowohl für 1 oder 2 Spieler gleichermaßen gut angebracht.
Das Programm hat, was man nicht von allen Oric-Spielen behaupten kann, ausgezeichnete Grafiken. Neun Schwierigkeitsstufen stehen zur Verfügung, wobei die schwersten kaum mehr zu bewältigen sind. PSS Programme sind auch in Deutschland im Fachhandel für einen angemessenen Preis zu erhalten.

Terror-Daktil für den Spectrum 48K

Die Story: Sie fliegen in einer kleinen alten Sportmaschine über die unerforschten südamerikanischen Dschungelgebiete.
Plötzlich gerät das Flugzeug außer Kontrolle. Die einzige Möglichkeit zur Notlandung ist am Fuße eines Vulkanes, weit ab von jeder Zivilisation. Als Sie zu sich kommen, ist es früher Morgen. Sie befinden sich auf einem Plateau, einige hundert Meter von dem Flugzeugwrack entfernt. Die Sonne geht auf und Sie schauen sich um: Da erblicken Sie einige merkwürdige, gefährlich aussehende Kreaturen – die Terror-Daktils. Und schon greifen diese Sie an. Ein Glück, daß Sie Ihre Pistole bei sich haben, mit der Sie versuchen müssen, die Monster in Schach zu halten.
Dieses in kurzen Worten zur Geschichte des Programms. Wir sind seit einiger Zeit dreidimensionale Spiele gewöhnt. Dieses hier wird als vierdimensional angekündigt und das ist es auch. Der Programmierer hat hier sehr gut die räumliche Distanz als Gestaltungsmerkmal eingesetzt. Es sieht schon sehr gut und professionell aus, wenn die Terror-Daktils beim Anflug immer größer werden, bis sie schließlich fast den ganzen Bildschirm ausfüllen.
Das Programm läuft entweder mit dem Kempston Joystickinterface, oder über die Tastatur. Etwas dürftig erscheint uns die akustische Untermalung des Programms, die für eine noch bessere Beurteilung notwendig gewesen wäre.

Kernbeißer

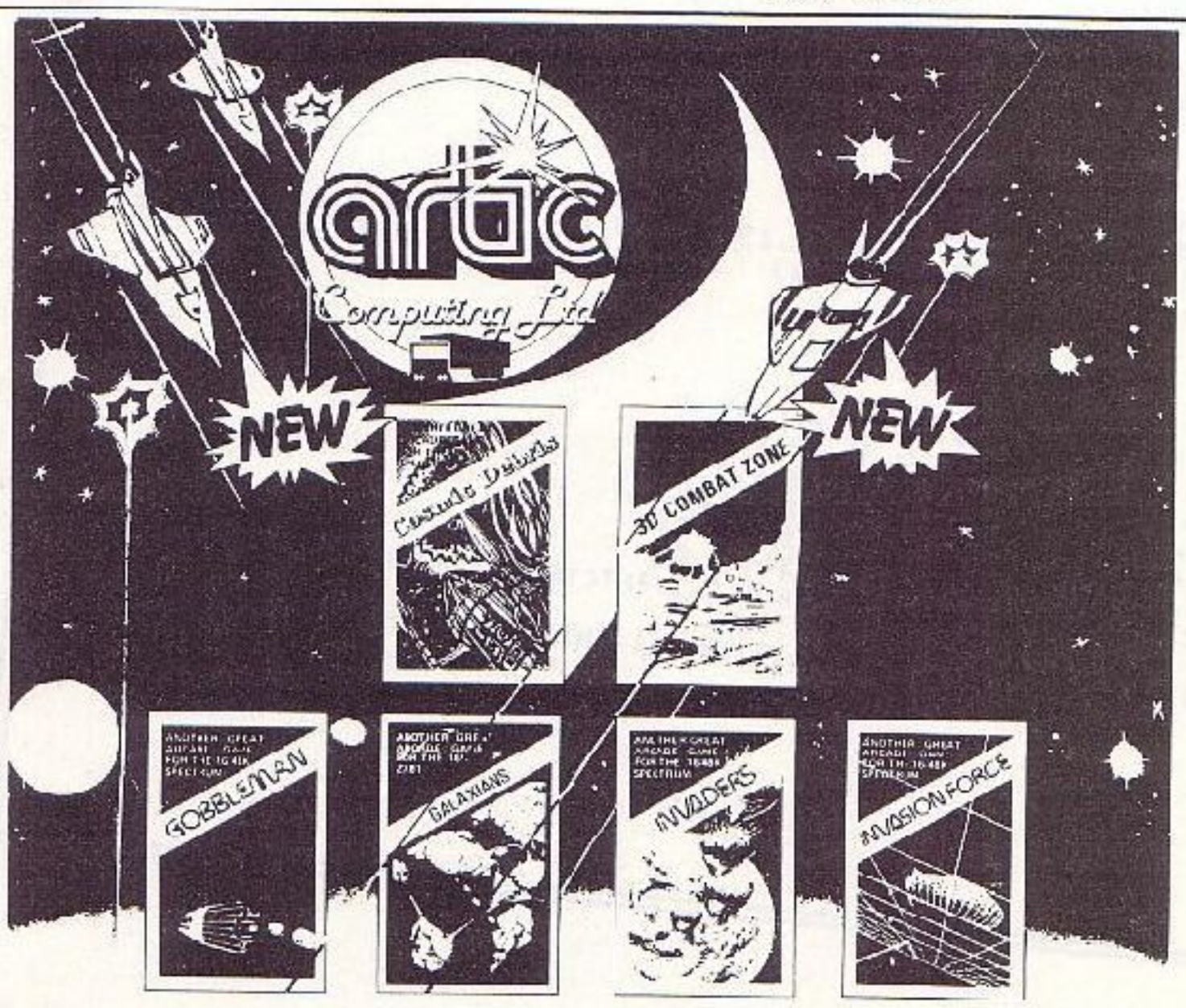
(Grundversion)

Die Aufgabe des Spiels besteht darin, einen vertikalen Tunnel, der von beiden Seiten durch eine Mauer gesichert ist, vor "Puc Man"-artigen Invadern zu verteidigen. Dies ist nur mit zwei Hämmern möglich, die rechts oder links aus einem Balken am Kopf der Mauer kommen, wenn man die Taste "M" bzw. "Z" drückt. Drückt man eine beliebige andere Taste, so fällt der Hammer nach unten und trifft eventuell einen Kernbeißer, der durch den Hammer betäubt zu Boden fällt und dort zerschmettert wird.
Wird er aber nicht getroffen, so erreicht er die Mauer und frißt die Steine. Die Kernbeißer greifen von rechts nach links und aus verschiedenen Höhenpositionen an, die jeweils vom Zufallsgenerator ermittelt werden.
Haben die Kernbeißer nun auf einer Höhe eine Reihe freigefressen, dann gelangt der nächste, der auf dieser Höhe kommt, in den Tunnel und dringt durch diesen in das innere Labyrinth ein.
Sie haben dem Angriff der Kernbeißer nicht standgehalten und die Runde ist beendet.
Wurde aber eine bestimmte Anzahl abgewehrt, sind Sie der Sieger.
Das Spiel hat drei verschiedene Angriffsstufen mit unterschiedlichem Schwierigkeitsgrad und Aussehen der Kernbeißer.
In der ersten Stufe sind sie ziemlich langsam und fressen nur einen Stein. In der zweiten am schnellsten und fressen gleich zwei Steine aus der Mauer. In der dritten Stufe sind die Kernbeißer zwar am langsamsten, doch sehr widerstandsfähig und noch gefräßiger, denn sie müssen zweimal getroffen werden und verschlingen gleich drei Steine.
Für jeden, außer Gefecht gesetzten Kernbeißer der ersten Stufe gibt es 10 Punkte, für die der zweiten 20 und für die der dritten 50 Punkte.
Score und Highscore werden während des Spieles angezeigt.
Der Programmaufbau ist in den REM-Zeilen erklärt.

```
100 CALL CLEAR
110 RANDOMIZE
120 REM  DEFINE CHARACTERS
130 CALL CHAR(104,"3C7EEED2C0CAF57F")
140 CALL CHAR(105,"3C7E774B0353AEFE")
150 CALL CHAR(112,"00FBFBFB00BFBFBF00")
160 CALL CHAR(113,"E0E0E0FFFFE0E040")
170 CALL CHAR(114,"070707FFFF070702")
180 CALL CHAR(136,"A96AAD5A3CFFFF7E")
190 CALL CHAR(115,"FFFFFFFFFFFFFFFF")
200 CALL CHAR(116,"FF0000000000000")
210 CALL CHAR(120,"3C66FEC0E0F0783E")
220 CALL CHAR(121,"3C667F03070F1E7C")
230 CALL CHAR(128,"3078CCFEBF908078")
240 CALL CHAR(129,"0C1E337FFF0D011E")
250 CALL CHAR(126,"3C4299A1A199423C")
260 CALL SCREEN(12)
270 CALL COLOR(10,13,12)
280 CALL COLOR(11,7,12)
290 CALL COLOR(12,2,12)
300 CALL COLOR(13,5,12)
310 REM   INSTRUCTIONS
320 PRINT TAB(10);"************";TAB(10);"*KERNBEISSER*";TAB(10);"************
"
330 PRINT
340 PRINT TAB(10);"ACTION";TAB(19);"KEYS"
350 PRINT
360 PRINT
370 PRINT "HAMMER RECHTS";TAB(19);"(M)"
380 PRINT
390 PRINT "HAMMER LINKS";TAB(19);"(Z)"
400 PRINT
410 PRINT "HAMMER DROP";TAB(19);"(ANY KEY)"
420 PRINT
430 PRINT
440 PRINT CHR$(126)
450 PRINT "BY BERNHARD PAHA"!!!!
460 CALL HCHAR(9,11,45,15)
470 CALL VCHAR(8,19,124,10)
480 PRINT "DRUECKEN SIE (S) ZUM START"
490 CALL KEY(0,KY,ST)
```

```
500 IF KY=83 THEN 510 ELSE 490
510 FOR DEL=1 TO 200
520 NEXT DEL
530 GOSUB 2020
540 PRI$="(HIGHSCORE:"&STR$(HS)&")"
550 X=16
560 Y=1
570 GOSUB 1880
580 STU=1
590 AZ=0
600 IF STU=4 THEN 2200
610 ON STU GOSUB 2780,2880,2980
620 REM   HAUPTPROGRAMM(BEWE-GUNG)
630 P=INT((23-6+1)*RND)+6
640 AZ=AZ+1
650 IF AZ=Z THEN 590
660 SS=INT(2*RND)+1
670 IF SS=2 THEN 950
680 REM    KERNBEISSER LINKS
690 FOR I=11 TO 15
700 CALL GCHAR(P,I,KL)
710 IF KL<>112 THEN 730
720 GOTO 740
730 NEXT I
740 L=I
750 A=L-11
760 FOR LS=1 TO 11 STEP S
770 CALL SOUND(50,294,2)
780 CALL HCHAR(P,LS,CH)
790 CALL KEY(0,KY,ST)
800 IF KY<>90 THEN 820
810 GOSUB 1250
820 CALL HCHAR(P,LS,32)
830 NEXT LS
840 A=A+11
850 IF SK=1 THEN 870
860 GOSUB 3080
870 FOR I=A TO L
880 IF I=16 THEN 2130
890 CALL SOUND(250,1000,0,110,1)
900 CALL HCHAR(P,I,CH)
910 CALL HCHAR(P,I,32)
920 NEXT I
930 T=0
940 GOTO 630
950 REM  KERNBEISSER RECHTS
960 FOR I=21 TO 17 STEP -1
970 CALL GCHAR(P,I,KR)
980 IF KR<>112 THEN 1000
990 GOTO 1010
1000 NEXT I
1010 L=I
1020 IF L=16 THEN 2130
1030 A=L-21
1040 FOR RS=32 TO 21 STEP -S
1050 CALL SOUND(50,294,2)
1060 CALL HCHAR(P,RS,CH1)
1070 CALL KEY(0,KY,ST)
1080 IF KY<>77 THEN 1100
1090 GOSUB 1610
1100 CALL HCHAR(P,RS,32)
1110 NEXT RS
1120 A=A+21
1130 IF SK=1 THEN 1150
1140 GOSUB 3140
1150 FOR I=A TO L STEP -1
1160 IF I=16 THEN 2130
1170 CALL SOUND(250,1000,0,110,1)
1180 CALL HCHAR(P,I,CH1)
1190 CALL HCHAR(P,I,32)
1200 NEXT I
1210 T=0
1220 GOTO 630
1230 END
1240 REM  ENDE DES HAUPTPRO- GRAMMES
1250 REM   BEWEGUNG HAMMER                LINKS
1260 FOR W=10 TO 1 STEP -1
1270 CALL HCHAR(5,W,113)
1280 CALL HCHAR(5,W,32)
1290 CALL KEY(0,KY,ST)
1300 IF ST=1 THEN 1340
1310 NEXT W
1320 RETURN
1330 REM  HAMMER LINKS DROP
1340 FOR F=5 TO P
1350 CALL VCHAR(F,W,113)
1360 CALL VCHAR(F,W,32)
1370 NEXT F
1380 CALL GCHAR(P,W,H)
1390 CALL GCHAR(P,LS,K)
1400 IF H=K THEN 1410 ELSE 1320
1410 IF SK<>3 THEN 1460
1420 IF T=1 THEN 1460
1430 T=1
1440 GOTO 1320
1450 REM   KERNB. LINKS DROP
1460 FOR I=P TO 23
1470 CALL HCHAR(I,W,CH)
1480 CALL HCHAR(I,W,32)
1490 NEXT I
1500 CALL COLOR(14,Y,12)
1510 CALL HCHAR(23,W,136)
1520 CALL SOUND(150,131,0,147,0,165,0)
1530 CALL HCHAR(23,W,32)
1540 T=0
1550 SC=SC+ADD
1560 PRI$="(SCORE: "&STR$(SC)&")"
1570 X=2
1580 Y=1
1590 GOSUB 1880
1600 GOTO 630
1610 REM   BEWEGUNG HAMMER                RECHTS
1620 FOR W=22 TO 32
1630 CALL HCHAR(5,W,114)
1640 CALL HCHAR(5,W,32)
1650 CALL KEY(0,KY,ST)
1660 IF ST=1 THEN 1690
1670 NEXT W
1680 RETURN
1690 REM  HAMMER RECHTS DROP
1700 FOR F=5 TO P
1710 CALL VCHAR(F,W,114)
1720 CALL VCHAR(F,W,32)
1730 NEXT F
1740 CALL GCHAR(P,W,H1)
1750 CALL GCHAR(P,RS,K1)
1760 IF H1=K1 THEN 1770 ELSE 1690
1770 IF SK<>3 THEN 1820
1780 IF T=1 THEN 1820
1790 T=1
1800 GOTO 1680
1810 REM  KERNB. RECHTS DROP
1820 FOR I=P TO 23
1830 CALL HCHAR(I,W,CH1)
1840 CALL HCHAR(I,W,32)
1850 NEXT I
1860 T=0
1870 GOTO 1500
1880 REM   PRINTROTINE
1890 FOR DRI=1 TO LEN(PRI$)
1900 CODE=ASC(SEG$(PRI$,DRI,1))
1910 GOSUB 1940
1920 NEXT DRI
1930 RETURN
1940 REM
1950 CALL HCHAR(Y,X,CODE)
1960 X=X+1
1970 RETURN
1980 REM    ERMITTLUNG HIGH-              SCORE
1990 IF SC<HS THEN 2010
2000 HS=SC
2010 RETURN
2020 REM  SCREEN SET UP
2030 CALL CLEAR
2040 FOR I=11 TO 15
2050 CALL VCHAR(6,I,112,18)
2060 NEXT I
2070 FOR I=17 TO 21
2080 CALL VCHAR(6,I,112,18)
2090 NEXT I
2100 CALL HCHAR(5,11,115,11)
2110 CALL HCHAR(24,1,116,32)
2120 RETURN
2130 REM  ENDE DES SPIELES
2140 CALL HCHAR(5,11,32,11)
2150 FOR I=P TO 5 STEP -1
2160 CALL VCHAR(I,16,CH)
2170 CALL VCHAR(I,16,32)
2180 NEXT I
```

```
2190 GOTO 2210
2200 GOSUB 2400
2210 PRI$='(SCORE: "&STR$(SC)&")'
2220 X=2
2230 Y=1
2240 GOSUB 1980
2250 GOSUB 1880
2260 PRI$='(HIGHSCORE: "&STR$(HS)&")"
2270 X=16
2280 Y=1
2290 GOSUB 1880
2300 PRI$="PLAY AGAIN? (J/N)"
2310 X=2
2320 Y=3
2330 GOSUB 1880
2340 SC=0
2350 AZ=0
2360 CALL KEY(0,KY,ST)
2370 IF KY=74 THEN 510
2380 IF KY=78 THEN 2390 ELSE 2360
2390 END
2400 REM  WINNER
2410 CALL CLEAR
2420 A=8
2430 X=6
2440 RESTORE 2740
2450 FOR I=1 TO 6
2460 A=A+2
2470 READ Z
2480 CALL HCHAR(A,X,Z)
2490 NEXT I
2500 IF M=1 THEN 2650
2510 RESTORE 2740
2520 B=8
2530 X1=8
2540 FOR I=1 TO 6
2550 B=B+2
2560 X1=X1+2
2570 READ Z
2580 CALL HCHAR(B,X1,Z)
2590 NEXT I
2600 X=25
2610 A=8
2620 M=1
2630 GOTO 2440
2640 RESTORE 2750
2650 FOR I=1 TO 5
2660 READ D,F,L
2670 CALL SOUND(D,F,L)
2680 NEXT I
2690 CALL SOUND(1000,110,0,-5,0,4000,0,500,0)
2700 FOR I=1 TO 30
2710 CALL SCREEN(5)
2720 CALL SCREEN(12)
2730 NEXT I
2740 DATA 87,73,78,78,69,82
2750 DATA 100,262,0,100,524,0,100,392,0,100,440,0,100,344,0
2760 CALL SCREEN(12)
2770 RETURN
2780 REM  SKILL LEVEL 1
2790 CH=120
2800 CH1=121
2810 SK=1
2820 ADD=10
2830 S=5
2840 STU=2
2850 V=2
2860 Z=20
2870 RETURN
2880 REM  SKILL LEVEL 2
2890 CH=128
2900 CH1=129
2910 SK=2
2920 ADD=20
2930 S=7
2940 STU=3
2950 V=5
2960 Z=15
2970 RETURN
2980 REM  SKILL LEVEL 3
2990 CH=104
3000 CH1=105
3010 SK=3
3020 ADD=50
3030 S=4
3040 STU=4
3050 V=13
3060 Z=12
3070 RETURN
3080 REM
3090 IF SK=3 THEN 3120
3100 L=L+1
3110 GOTO 3130
3120 L=L+2
3130 RETURN
3140 REM
3150 IF SK=3 THEN 3180
3160 L=L-1
3170 GOTO 3190
3180 L=L-2
3190 RETURN
```



Bestellungen an:

3443 Herleshausen
Nordstraße 22

WICOSOFT

Bitte benutzen Sie Bestellkarte

Digger

für den TI-99/4A mit Extended Modul

Das Spiel "Digger" ist von seinem Ablauf her dem bekannten Pac Man ähnlich.

In einem Labyinth muß man mit dem Hauptdarsteller des Spieles – Digger – genannt (gelbe Figur), drei Monster (als blaue Figuren dargestellt) jagen und von der Bildfläche verschwinden lassen. Dies geht aber nur, wenn zuvor die roten Kraftpillen, die im Labyrinth verteilt sind, vorher verspeist wurden.
Hat man ein Monster vernichtet, so zähl: dies 50 Punkte.
Es gibt außerdem noch grüne Kraftpillen, mit denen man die Mauern durchfahren kann und weiße "Superkraftpillen", mit denen man fahren kann wohin man will.
Mit jeder der Pillen ist es möglich, seinen eigenen, schwarz markierten Wegen nachzugehen.
Die eigentliche Aufgabe des Spieles ist nicht die Monster zu jagen, sondern sämtliche im Labyrinth ausgelegten Herzen zu sammeln und sich dabei nicht festzufahren.
Gesteuert wird Digger über die Tastatur:
E = oben, X = unten, S = links, D = rechts, ENTER = aufgeben.

Erläuterung der Zeilen:

```
100 - 230  Farbanweisungen und
           Spielvorbereitung.
240 - 370  1. Titelbild
380 - 420  2. Titelbild
430 - 470  Labyrinthaufbau
480 - 660  Pillen und Monster werden ins Labyrinth gelegt.
670 - 850  Key Abfrage
860 - 1070 Monstersteuerung
1080 - 1190 Ende
1200 - 1260 Data-Blöcke
1270 - 2160 Gchar Unterprogramm
2170 - 2190 Herzchen Unterprogramm
```

```
!********
!*DIGGER*
!********
!
!**********************
! *GUENTER LIESNER*
!**********************
100 DIM C(3),D(3)
110 CALL MAGNIFY(2):: CALL SCREEN(2)
120 CALL CLEAR
130 CALL CHAR(136,"183C7E99997E427E")
140 CALL CHAR(129,"3C7EDBDBDBFFC3BD")
150 CALL CHAR(120,"DFDFA7008100C700")
160 S$="003C5A666E5A3C00"
170 CALL CHAR(33,S$)
180 CALL CHAR(40,S$)
190 CALL CHAR(112,S$)
200 CALL CHAR(104,"00E7998142241818")
210 CALL CHAR(96,"AA55AA55AA55AA55")
220 CALL COLOR(9,7,2,10,9,1,11,15,1,12,3,1,13,5,1,14,11,1,1,3,1,2,9,1)
230 CALL COLOR(3,13,1):: CALL COLOR(4,13,1):: FOR Z1=5 TO 8 :: CALL COLOR(Z1,16,
1):: NEXT Z1
231 !**************
232 ! *1.TITELBILD*
233 !**************
240 CALL SPRITE(#1,68,INT(13*RND)+3,1,64,1,0)
250 CALL SPRITE(#2,73,INT(13*RND)+3,30,80,1,0)
260 CALL SPRITE(#3,71,INT(13*RND)+3,60,96,1,0)
270 CALL SPRITE(#4,71,INT(13*RND)+3,90,112,1,0)
280 CALL SPRITE(#5,69,INT(13*RND)+3,120,128,1,0)
290 CALL SPRITE(#6,82,INT(13*RND)+3,150,144,1,0)
300 FOR Z1=7 TO 12 :: CALL SPRITE(#Z1,104,9,1,1,INT(100*RND)-50,INT(100*RND)-50)
:: NEXT Z1
310 FOR Z1=13 TO 18 :: CALL SPRITE(#Z1,112,15,1,1,INT(100*RND)-50,INT(100*RND)-5
0):: NEXT Z1
```

```
320 FOR Z1=19 TO 23 :: CALL SPRITE(#Z1,137,11,1,1,INT(100*RND)-50,INT(100*RND)-5
0):: NEXT Z1
330 FOR Z1=24 TO 28 :: CALL SPRITE(#Z1,129,5,1,1,INT(100*RND)-50,INT(100*RND)-50
):: NEXT Z1
340 FOR Z1=5 TO 125 STEP 5 :: CALL MOTION(#1,Z1,0,#2,Z1,0,#3,Z1,0,#4,Z1,0,#5,Z1,
0,#6,Z1,0)
360 NEXT Z1
370 FOR Z2=1 TO 28 :: CALL DELSPRITE(#Z2):: NEXT Z2
371 ! ************
372 ! *2.TITELBILD*
373 ! ************
380 DISPLAY AT(5,3):"PRESS ANY KEY TO BEGIN" :: DISPLAY AT(7,7):"GUENTER LIESNER
" :: DISPLAY AT(9,7):"SOFTWARE 1984"
390 DISPLAY AT(24,13):"DIGGER"
400 CALL SPRITE(#1,137,11,192/2,256/2):: FOR Z1=1 TO 200 :: NEXT Z1 :: FOR Z1=1
TO 6 :: CALL SOUND(250,-Z1,0):: CALL KEY(0,A5,AD):: IF AD<>0 THEN 420
410 NEXT Z1 :: GOTO 400
420 CALL DELSPRITE(#1)
421 ! ****************
422 ! *LABYRINDAUFBAU*
423 ! ****************
430 FOR Z1=1 TO 14 :: CALL COLOR(Z1,1,1):: NEXT Z1 :: CALL CLEAR :: RESTORE :: F
OR Z1=1 TO 22 :: READ A$ :: PRINT A$;:: NEXT Z1
440 CALL COLOR(9,7,2,10,9,1,11,15,1,12,3,1,13,5,1,14,11,1,1,3,1,2,9,1)
450 CALL COLOR(3,13,1):: CALL COLOR(4,13,1):: FOR Z1=5 TO 8 :: CALL COLOR(Z1,16,
1):: NEXT Z1
460 DISPLAY AT(1,1):"SCORE: 0";
470 DISPLAY AT(1,14):"HI=SCORE:";HISCORE
471 ! ********************
472 ! *PILLEN UND MONSTER*
473 ! ********************
480 RANDOMIZE
490 FOR Z1=1 TO 10
500 Z2=INT(22*RND)+3 :: Z3=INT(28*RND)+3
510 CALL GCHAR(Z2,Z3,Z4):: IF Z4<>96 THEN 500
520 CALL HCHAR(Z2,Z3,104):: NEXT Z1
530 FOR Z1=1 TO 3
540 Z2=INT(22*RND)+3 :: Z3=INT(28*RND)+3 :: CALL GCHAR(Z2,Z3,Z4):: IF Z4<>96 THE
N 540 ELSE CALL HCHAR(Z2,Z3,33)
550 NEXT Z1
560 FOR Z1=1 TO 3
570 Z2=INT(22*RND)+3 :: Z3=INT(28*RND)+3 :: CALL GCHAR(Z2,Z3,Z4):: IF Z4<>96 THE
N 570 ELSE CALL HCHAR(Z2,Z3,40)
580 NEXT Z1
590 Z1=INT(22*RND)+3 :: Z2=INT(28*RND)+3 :: CALL GCHAR(Z1,Z2,Z3):: IF Z3<>96 THE
N 590 ELSE CALL HCHAR(Z1,Z2,112)
600 A=INT(22*RND)+3 :: B=INT(28*RND)+3
610 CALL GCHAR(A,B,Z1):: IF Z1<>96 THEN 600 ELSE CALL HCHAR(A,B,107)
620 FOR Z1=1 TO 3
630 Z2=INT(22*RND)+3 :: Z3=INT(28*RND)+3
640 CALL GCHAR(Z2,Z3,Z4):: IF Z4<>95 THEN 630
650 C(Z1)=Z2 :: D(Z1)=Z3 :: CALL HCHAR(C(Z1),D(Z1),123)
660 NEXT Z1
661 ! *****
662 ! *KEY*
663 ! *****
670 CALL KEY(0,K,S)
680 E=A :: F=B
690 GLUECK=(100*RND)+1 :: IF GLUECK<99 THEN 740
700 GLUECK=INT(GLUECK)
710 PILLEN=INT(2*RND)+0 :: ZEICHEN=33+(PILLEN*7):: XPOSITION=INT(22*RND)+3 :: YP
OSITION=INT(28*RND)+3 :: CALL GCHAR(XPOSITION,YPOSITION,AZ)
720 IF AZ<>96 THEN 710 ELSE 730
730 CALL HCHAR(XPOSITION,YPOSITION,ZEICHEN):: CALL SOUND(1000,3000+(PILLEN*1000)
,0,2000+(PILLEN*1000),2,1000+(PILLEN*1000),3)
740 A=A-(K=88)+(K=69)
750 B=B-(K=68)+(K=83)
760 IF A=E AND B=F THEN 770 ELSE 810
770 SCORE=SCORE-30
780 IF K<>84 THEN 800 ELSE DISPLAY AT(2,4)BEEP "PRESS R TO RETURN"
790 CALL KEY(0,K9,S9):: IF K9<>ASC("R")THEN 790 ELSE CALL HCHAR(2,1,32,32)
800 IF K=13 THEN 1080
810 CALL GCHAR(A,B,M)
820 DISPLAY AT(1,7):SCORE;
830 IF M=137 THEN M=96
840 GOSUB 1270
850 GOTO 860
851 ! **********************
852 ! *MONSTERSTEUERUNG*
853 ! **********************
860 Z1=INT(3*RND)+1
```

```
870 X=C(Z1):: Y=D(Z1)
880 P(1)=MAX(A,C(Z1)):: P(2)=MIN(B,D(Z1)):: P(3)=MAX(B,D(Z1)):: P(4)=MIN(A,C(Z1
)
890 O(1)=P(1)-P(4)
900 O(2)=P(3)-P(2)
910 IF O(1)=0 AND O(2)=0 THEN 1080
920 IF O(1)>O(2)THEN 960 ELSE 930
930 IF O(1)=0 THEN 960
940 IF A>C(Z1)THEN C(Z1)=C(Z1)+1 ELSE C(Z1)=C(Z1)-1
950 GOTO 980
960 IF O(2)=0 THEN 930
970 IF B>D(Z1)THEN D(Z1)=D(Z1)+1 ELSE D(Z1)=D(Z1)-1
980 CALL GCHAR(C(Z1),D(Z1),G)
990 IF G=96 OR G=32 THEN 1050
1000 IF G=137 AND COLOR=0 THEN 1090
1010 IF G<>120 THEN 1030
1020 C(Z1)=X :: D(Z1)=Y :: GOTO 1050
1030 IF G=137 AND COLOR=1 THEN 1090
1040 GOTO 1020
1050 CALL HCHAR(X,Y,96)
1060 CALL HCHAR(C(Z1),D(Z1),129)
1070 GOTO 670
1071 ! ******
1072 ! *ENDE*
1073 ! ******
1080 CALL SOUND(4000,-8,0):: CALL CLEAR
1090 CALL SOUND(100,20000,0):: CALL CLEAR
1100 HISCORE=MAX(SCORE,HISCORE):: IF HISCORE=SCORE THEN DISPLAY AT(12,2):"NEUER
HIGHSCORE:";HISCORE ELSE DISPLAY AT(12,2):"IHRE PUNKTZAHL:";SCORE
1110 DISPLAY AT(24,2):"NEUES SPIEL <J> ODER <N>?"
1120 CALL SPRITE(#18,129,5,50,70,INT(250*RND)-127,INT(250*RND)-127)
1130 CALL SPRITE(#19,137,11,100,130,INT(250*RND)-127,INT(250*RND)-127)
1140 FOR Z1=1 TO 3 :: CALL SPRITE(#(Z1+19),112,11,INT(196*RND)+1,INT(250*RND)+1,
INT(100*RND)-50,INT(100*RND)-50)
1150 CALL SPRITE(#(Z1+23),104,9,INT(196*RND)+1,INT(250*RND)+1,INT(100*RND)-50,IN
T(100*RND)-50):: NEXT Z1
1160 CALL KEY(0,K6,S6):: IF S6=0 THEN 1160 :: IF K6=74 THEN 1170 ELSE 1180
1170 CALL DELSPRITE(ALL):: CALL CLEAR :: COLOR,HERZ,SCORE=0 :: GOTO 100
1180 IF K6=78 THEN END
1190 GOTO 1160
1191 ! ******
1192 ! *DATA*
1193 ! ******
1200 DATA XXXXXXXXXXXXXXXXXXXXXXXXXXXX,X`X`````XXXXXXXXXX`````X`X,X`X`X`````
`````````X`X`X,X`````X`````XXXX```XXXX```X`X`X
1210 DATA X```X`X`X`````````X`X`X```X,X`X`X`X`````````X`X`X`X,X`````X```XX
1220 DATA X```X`X`X```XXXX```X`X`X```X,X`X`X`X`````````X`X`X`X,X`````X```X`
X```XXX```X```X`X
1230 DATA X`X`X`X`````````X`X`X`X,X```X`X```XXXX`X```X`X`X`X`X`X`X`X`X`X`X`X`
XXXX`X`````X```X
1240 DATA X`````X`````X`XXXX`X```X`X`X,X```X`X```X`XXXX`X```X`X`X`X`X`X`X`X`
`````X`X`X`X,X`X```X```XXX```XXX```X`X`X,X`X`X`X```XXXX```X`X`X```X,X`X`X`X`
1250 DATA X`X`X`X`X`````````X`X`X`X,X`X`````XXXXXXXXXX`````X`X,XXXXXXXXXXX
`````X`X`X`X,X```X```XXXX```XXXX```X`X`X
1260 DATA X`X`X`X`````````X`X`X`X,X`X`````XXXXXXXXXX`````X`X,XXXXXXXXXXX
XXXXXXXXXXXXXXX
1261 ! ************
1262 ! *GCHAR(SUB)*
1263 ! ************
1270 IF COLOR<>0 THEN 1430 ELSE Q=32
1280 IF M<>96 THEN 1300
1290 SCORE=SCORE+5 :: GOTO 2150
1300 IF M<>120 THEN 1320
1310 A=E :: B=F :: CALL SCREEN(9):: CALL SOUND(250,-3,0):: CALL SCREEN(2):: GOTO
 2150
1320 IF M<>32 THEN 1340
1330 A=E :: B=F :: CALL SOUND(250,-1,0):: GOTO 2150
1340 IF M=129 THEN 1090
1350 IF M<>33 THEN 1370
1360 COLOR=1 :: CALL COLOR(14,3,1):: SCH1=20 :: GOTO 2150
1370 IF M<>40 THEN 1390
1380 COLOR=2 :: CALL COLOR(14,9,1):: SCH2=20 :: GOTO 2150
1390 IF M<>104 THEN 1420
1400 HERZ=HERZ+1 :: SCORE=SCORE+100 :: CALL SOUND(500,262,0,330,0,392,0)
1410 IF HERZ=10 THEN 2170 ELSE 2150
1420 COLOR=11 :: CALL COLOR(14,16,1):: SCH3=20 :: GOTO 2150
1430 IF COLOR<>1 THEN 1650 ELSE Q=96
1440 SCH1=SCH1-1 :: IF SCH1<6 THEN CALL COLOR(14,8,1)
1450 IF SCH1<>0 THEN 1470
1460 COLOR=0 :: CALL COLOR(14,11,1)
1470 IF M<>96 THEN 1490
```
```

```
1480 SCORE=SCORE+5 :: GOTO 2150
1490 IF M<>120 THEN 1510
1500 CALL SOUND(250,-3,0):: SCORE=SCORE+25 :: GOTO 2150
1510 IF M=129 THEN 1090
1520 IF M<>32 THEN 1540
1530 SCORE=SCORE+10 : CALL SOUND(250,-3,0):: GOTO 2150
1540 IF M<>104 THEN 1570
1550 HERZ=HERZ+1 :: SCORE=SCORE+100 :: CALL SOUND(500,262,0,330,0,392,0)
1560 IF HERZ=10 THEN 2170 ELSE 2150
1570 IF M<>33 THEN 1590
1580 A=E :: B=F :: CALL SOUND(100,1000,0):: GOTO 2150
1590 IF M<>40 THEN 1640
1600 COLOR=2 :: CALL COLOR(14,9,1):: SCH2=20 :: GOTO 2150
1610 IF M<>104 THEN 1640
1620 HERZ=HERZ+1 :: SCORE=SCORE+100 :: CALL SOUND(500,262,0,330,0,392,0)
1630 IF HERZ=10 THEN 2170
1640 COLOR=11 :: CALL COLOR(14,16,1):: SCH3=20 :: GOTO 2150
1650 IF COLOR<>2 THEN 1900 ELSE Q=96
1660 SCH2=SCH2-1 :: IF SCH2<6 THEN CALL COLOR(14,8,1)
1670 IF SCH2<>0 THEN 1690
1680 COLOR=0 :: CALL COLOR(14,11,1)
1690 IF M<>96 THEN 1710
1700 SCORE=SCORE+5 :: GOTO 2150
1710 IF M<>120 THEN 1730
1720 A=E :: B=F :: CALL SCREEN(9):: CALL SOUND(2 FÄÄÄ-3,0):: CALL SCREEN(2):: GOTO 2150
1730 IF M<>32 THEN 1750
1740 CALL SOUND(250,-1,0):: GOTO 2150
1750 IF M<>129 THEN 1820
1760 FOR Z1=1 TO 8 :: CALL SOUND(-90,-Z1,0):: NEXT Z1
1770 SCORE=SCORE+50 :: FOR Z1=1 TO 3
1780 IF A=C(Z1)AND B=D(Z1)THEN 1810
1790 NEXT Z1
1800 CALL HCHAR(C(Z1),D(Z1),96)
1810 C(Z1)=INT(22*RND)+3 :: D(Z1)=INT(28*RND)+3 :: CALL GCHAR(C(Z1),D(Z1),Z4)::
IF Z4<>96 THEN 1810 ELSE CALL HCHAR(C(Z1),D(Z1),129):: GOTO 2150
1820 IF M<>33 THEN 1840
1830 COLOR=1 :: CALL COLOR(14,3,1):: SCH1=20 :: GOTO 2150
1840 IF M<>40 THEN 1860
1850 A=E :: B=F :: CALL SOUND(100,1000,0):: GOTO 2150
1860 IF M<>104 THEN 1890
1870 HERZ=HERZ+1 :: SCORE=SCORE+100 :: CALL SOUND(500,262,0,330,0,392,0)
1880 IF HERZ=10 THEN 2170 ELSE 2150
1890 COLOR=11 :: CALL COLOR(14,16,1):: SCH3=20 :: GOTO 2150
1900 Q=96
1910 SCH3=SCH3-1 :: IF SCH3<6 THEN CALL COLOR(14,8,1)
1920 IF SCH3<>0 THEN 1940
1930 COLOR=0 :: CALL COLOR(14,11,1)
1940 IF M<>96 THEN 1960
1950 SCORE=SCORE+5 :: GOTO 2150
1960 IF M<>120 THEN 1980
1970 CALL SOUND(250,-3,0):: GOTO 2150
1980 IF M<>32 THEN 2000
1990 CALL SOUND(250,-1,0):: GOTO 2150
2000 IF M<>129 THEN 2070
2010 FOR Z1=1 TO 8 :: CALL SOUND(-90,-Z1,0):: NEXT Z1
2020 SCORE=SCORE+50 :: FOR Z1=1 TO 3
2030 IF A=C(Z1)AND B=D(Z1)THEN 2060
2040 NEXT Z1
2050 CALL HCHAR(C(Z1),D(Z1),96)
2060 C(Z1)=INT(22*RND)+3 :: D(Z1)=INT(28*RND)+3 :: CALL GCHAR(C(Z1),D(Z1),Z4)::
IF Z4<>96 THEN 1810 ELSE CALL HCHAR(C(Z1),D(Z1),129):: GOTO 2150
2070 IF M<>33 THEN 2090
2080 COLOR=1 :: CALL COLOR(14,3,1):: SCH1=20 :: GOTO 2150
2090 IF M<>40 THEN 2110
2100 A=E :: B=F :: CALL SOUND(100,1000,0):: GOTO 2150
2110 IF M<>104 THEN 2140
2120 HERZ=HERZ+1 :: SCORE=SCORE+100 :: CALL SOUND(500,252,0,330,0,392,0)
2130 IF HERZ=10 THEN 2170 ELSE 2150
2140 CALL SOUND(100,1000,0):: A=E :: B=F :: GOTO 2150
2150 CALL HCHAR(E,F,Q):: CALL HCHAR(A,B,137)
2150 RETURN
2161 ! **********
2162 ! *HERZ(SUB)*
2163 ! **********
2170 FOR Z1=1 TO 10 :: CALL SOUND(100,Z1*100+110*2,0):: CALL SOUND(100,Z1*200+11
0*3,0)
2180 Z2=INT(22*RND)+3 :: Z3=INT(28*RND)+3 :: CALL GCHAR(Z2,Z3,Z4):: IF Z4<>96 TH
EN 2180 ELSE CALL HCHAR(Z2,Z3,104)
2190 NEXT Z1 :: HERZ=0 :: GOTO 2150
```

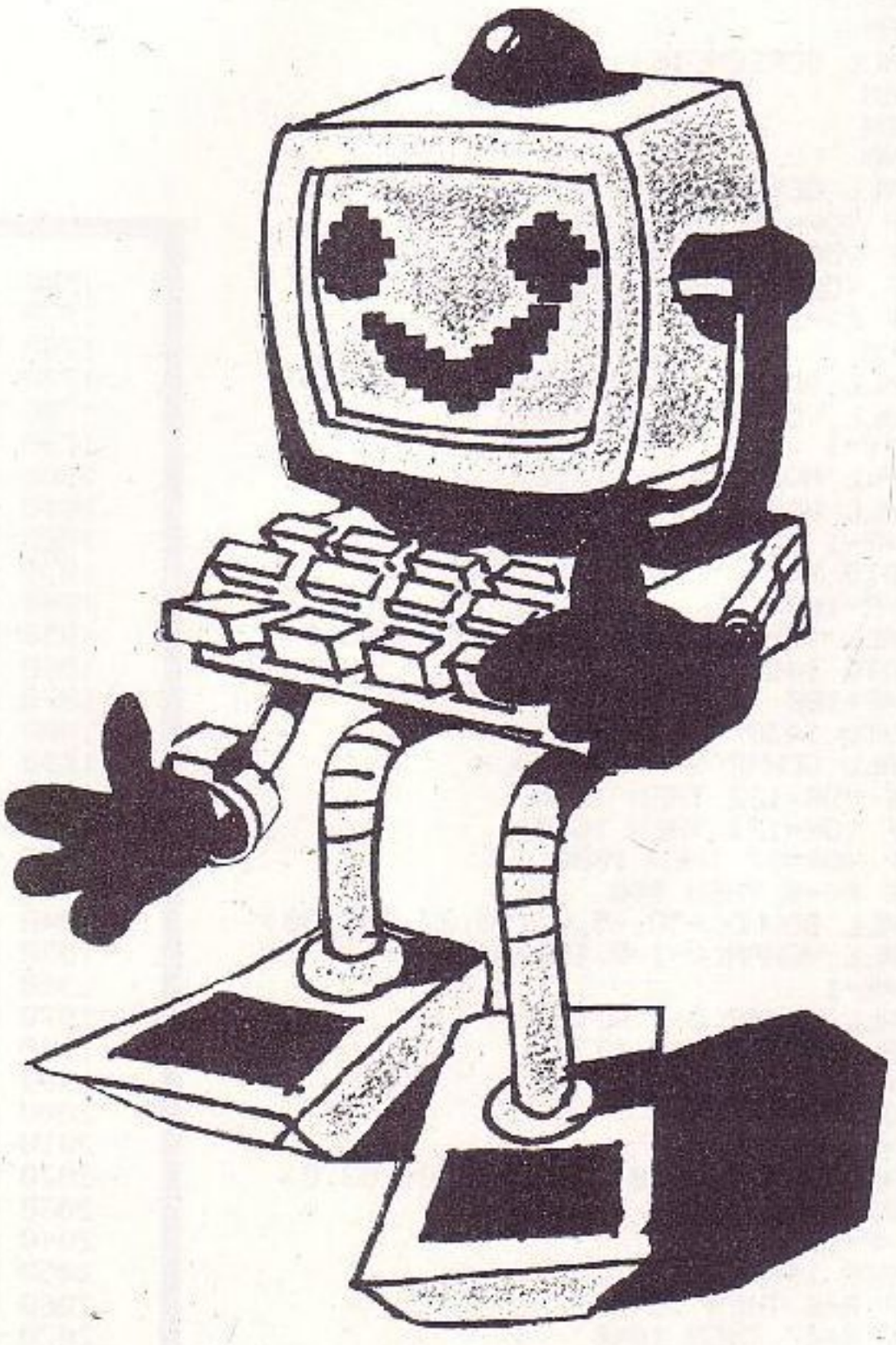
Help

Bei diesem Spiel geht es darum, einen Computer, der die Menschheit bedroht, auszuschalten. Dabei wird man von Robotern behindert, während man versucht, die Lampen des Computers auszuschalten und die Roboter damit zu zerstören.
Bei Beginn und Ende des Programmes, wird die 9. Sinfonie von Beethoven gespielt.
Ein recht gelungenes Spiel mit viel Grafik und Musik. Gesteuert wird mit den Cursortasten, wobei die Spacetaste zum Zerstören des Roboters dient.

```
100 DIM ZU(50),FALL(50)
110 RANDOMIZE
120 CALL SCREEN(1)
130 GOSUB 2930
140 CALL SCREEN(2)
150 PRINT "NAME DES SPIELERS:";NAME$
160 PRINT
170 PRINT " 01                                    01  "
180 PRINT " 01                                    01  "
190 PRINT " 01                                    01  "
200 PRINT " 01                                    01  "
210 PRINT " 01                                    01  "
220 PRINT " 01                                    01  "
230 PRINT " 01                                    01  "
240 PRINT "aaa  aaaaaaaaaaaaaaaaaa                aaa"
250 PRINT "  h                                    i  "
260 PRINT "  h                                    i  "
270 PRINT "  h    PPPPPP  PPPPPP                  i  "
280 PRINT "  h    P(,,*P  P(,,*P                  i  "
290 PRINT "  h    P,,,,P  P,,,,P                  i  "
300 PRINT "  h    P),,+P  P),,+P                  i  "
310 PRINT "  h    PPPPPP  PPPPPP                  i  "
320 PRINT "  h                                    i  "
330 PRINT "  h                                    i  "
340 PRINT "  h                                    i  "
350 PRINT "  h                                    i  "
360 PRINT "  h                                    i  "
370 PRINT "  h                                    i  "
380 PRINT "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
390 FOR I=1 TO 12
400 CALL COLOR(12,16,1)
410 NEXT I
420 CALL CHAR(97,"FFFFFFFFFFFFFFFF")
430 Q=22
440 R=14
450 CALL HCHAR(Q-1,R,112)
460 CALL HCHAR(Q,R,113)
470 L=1
480 RI=1
490 CALL CHAR(150,"FFFFFFFFFFFFFFFF")
500 CALL COLOR(16,1,1)
510 P=5000
520 CALL CHAR(44,"0")
530 CALL CHAR(112,"7C547C4C38107CBA")
540 CALL CHAR(113,"BABABA2828282E6C")
550 CALL CHAR(120,"7E5A7E427E187E7E")
560 CALL CHAR(121,"7E7E7E7E7C6C6C6C")
570 CALL CHAR(122,"00183C7E7E3C1800")
580 CALL CHAR(128,"3C7EFFFFFFFF7E3C")
590 CALL CHAR(40,"FFFCF0E0C0C08080")
600 CALL CHAR(41,"8080C0C0E0F0FCFF")
610 CALL CHAR(42,"FF3F0F0703030101")
620 CALL CHAR(43,"0101030307070F3FFF")
630 CALL CHAR(48,"C0C0C0FFC0C0C0C0")
640 CALL CHAR(49,"030303FF03030303")
650 CALL CHAR(56,"00402001070F0FDF")
660 CALL CHAR(57,"80840880E0F0F0FB")
670 CALL CHAR(58,"0F0F07011020")
680 CALL CHAR(59,"F0F0E01008048080")
690 CALL CHAR(104,"0202020202020202")
700 CALL CHAR(105,"4040404040404040")
710 CALL CHAR(72,"00183C7E7E3C1800")
720 CALL CHAR(80,"FFFFFFFFFFFFFFFFFFFF")
730 CALL COLOR(11,16,1)
740 CALL COLOR(12,15,1)
750 CALL COLOR(13,9,1)
760 CALL COLOR(4,12,1)
770 CALL COLOR(3,10,1)
780 CALL COLOR(2,10,6)
790 CALL COLOR(10,8,1)
800 CALL COLOR(6,14,1)
810 CALL COLOR(7,10,1)
820 CALL COLOR(9,14,1)
830 FOR SOU=1 TO 62
840 READ ABC,DEFZ,GHI,JKL
850 CALL SOUND(INT(JKL*1.5),ABC,0,DEFZ,0,GHI,0)
860 NEXT SOU
870 FOR I=1 TO 50
880 RANDOMIZE
890 ZU(I)=INT(RND*14)+1
900 FALL(I)=INT(RND*28)+1
910 NEXT I
920 CALL KEY(0,X,Y)
930 IF X=68 THEN 1360
940 IF X=83 THEN 1540
950 IF X=69 THEN 1710
960 IF X=88 THEN 2790
980 IF X=32 THEN 1950
```

```
1000 NOW=NOW+ZU(RI)
1010 NI=NI+1
1020 FAU=FAU+1
1030 NEU=NEU+1
1040 IF RI>50 THEN 890
1050 IF FAU=ZU(RI)THEN 2010
1060 IF I>50 THEN 890
1070 IF NEU=FALL(L)THEN 2310
1080 IF P<=0 THEN 1120
1090 P=P-1
1110 GOTO 920
1120 CALL CLEAR
1125 RESTORE 3340
1130 FOR SOU=1 TO 62
1140 X=INT(RND*16)+1
1150 CALL SCREEN(X)
1160 READ ABC,DEFZ,GHI,JKL
1170 CALL SOUND(INT(JKL*1.3),ABC,0,DEFZ,0,GHI,0)
1180 NEXT SOU
1190 CALL CLEAR
1200 RANDOMIZE
1210 NOW=INT(RND*100)+NOW
1220 GOSUB 3450
1230 PRINT "LEIDER IST DAS SPIEL JETZT  ZUENDE.                    SIE MACHTEN"
;NOW;"SCHRITTE "
1240 PRINT "UND RETTETEN SOMIT";NOW
1250 PRINT "MENSCHENLEBEN.":::
1260 PRINT "DIE HOECHSTE PUNKTZAHL      BISHER BETRUG:";HIGH
1270 PRINT " BRAVO,";NAME$;"!"
1280 REM
1290 REM
1300 REM
1310 REM
1320 CALL SCREEN(16)
1330 REM
1340 REM
1350 END
1360 CALL GCHAR(Q,R+1,VOR)
1370 IF VOR=122 THEN 1490
1380 IF VOR=121 THEN 1490
1390 IF VOR=97 THEN 1520
1400 IF K>=27 THEN 940
1410 REM
1420 CALL SOUND(30,-5,0,110,30,150,30)
1430 CALL VCHAR(Q-1,R,150,2)
1440 R=R+1
1450 CALL HCHAR(Q-1,R,112)
1460 CALL HCHAR(Q,R,113)
1470 P=P-1
1480 GOTO 940
1490 P=P-1000
1500 CALL SOUND(-200,-5,0,200,0,205,0)
1510 GOTO 1430
1520 P=P+100
1530 GOTO 1430
1540 CALL GCHAR(Q,R-1,VOR)
1550 IF VOR=122 THEN 1660
1560 IF VOR=121 THEN 1660
1570 IF VOR=97 THEN 1690
1580 IF R<=6 THEN 950
1590 CALL SOUND(-30,-5,0,110,27,155,30)
1600 CALL VCHAR(Q-1,R,150,2)
1610 R=R-1
1620 CALL HCHAR(Q-1,R,112)
1630 CALL HCHAR(Q,R,113)
1640 P=P-1
1650 GOTO 950
1660 P=P-1000
1670 CALL SOUND(-200,-5,0,200,0,203,0)
1680 GOTO 1600
1690 P=P+100
1700 GOTO 1600
1710 IF R=5 THEN 1840
1720 IF R=27 THEN 1840
1730 CALL GCHAR(Q-2,R,UEBER)
1740 IF UEBER=58 THEN 1780
1750 IF UEBER=59 THEN 1780
1760 P=P-100
1770 GOTO 960
1780 CALL HCHAR(Q-2,R-1,150,3)
1790 CALL HCHAR(Q-3,R-1,150,3)
1800 LA=LA-1
1810 CALL SOUND(-900,1500,0,1504,1,1507,3)
1820 P=P+500
1830 GOTO 960
1840 IF Q<>9 THEN 1880
1850 P=P-100
1860 GOTO 960
1870 CALL SOUND(-300,-6,0,500,23,800,28)
1880 CALL VCHAR(Q-1,R,150,2)
1890 CALL SOUND(-100,1000,0,1001,0,1001,0)
1900 Q=8
1910 CALL HCHAR(Q-1,R,112)
1920 CALL HCHAR(Q,R,113)
1930 P=P+100
1940 GOTO 960
1950 CALL SOUND(-200,110,0,800,0,2500,0)
1960 CALL VCHAR(Q-1,R-1,150,2)
1970 CALL VCHAR(Q-1,R+1,150,2)
1980 P=P-100
1990 CALL SOUND(-200,110,0,800,0,2500,0)
2000 GOTO 1050
2010 RI=RI+1
2020 FAU=0
2030 CALL SOUND(-250,-4,10,110,0,113,0)
2040 RANDOMIZE
2050 AB=INT(RND*1)+1
2060 RANDOMIZE
2070 CD=INT(RND*2)+1
2080 IF AB>1 THEN 2190
```

```
2090 IF CD<>1 THEN 2130
2100 CALL HCHAR(Q-1,R-1,120)
2110 CALL HCHAR(Q,R-1,121)
2120 GOTO 1070
2130 IF CD=2 THEN 2170
2140 CALL HCHAR(Q,R-1,97)
2150 REM
2160 GOTO 1070
2170 CALL HCHAR(Q,R-1,122)
2180 GOTO 1070
2190 IF CD<>1 THEN 2230
2200 CALL HCHAR(Q-1,R+1,120)
2210 CALL HCHAR(Q,R+1,121)
2220 GOTO 1070
2230 IF CD<>2 THEN 2280
2240 Q=22
2250 R=14
2260 CALL HCHAR(Q,R+1,122)
2270 GOTO 1070
2280 CALL HCHAR(Q,R+1,97)
2290 GOTO 1070
2300 RAHDOMIZE
2310 ZUF=INT(RND*11)+1
2320 NEU=0
2330 ON ZUF GOSUB 2430,2460,2490,2520,2550,2580,2610,2640,2670,2700,2730,2760
2340 CALL HCHAR(POS1,POS2,56)
2350 CALL HCHAR(POS1,POS2+1,57)
2360 CALL HCHAR(POS1+1,POS2,58)
2370 CALL HCHAR(POS1+1,POS2+1,59)
2380 CALL SOUND(-300,-1,9,2000,0,2010,0)
2390 LA=LA+1
2400 IF LA<3 THEN 2420
2410 P=P-1000
2420 GOTO 1090
2430 POS1=5
2440 POS2=7
2450 RETURN
2460 POS1=5
2470 POS2=10
2480 RETURN
2490 POS1=5
2500 POS2=13
2510 RETURN
2520 POS1=5
2530 POS2=16
2540 RETURN
2550 POS1=5
2560 POS2=19
2570 RETURN
2580 POS1=5
2590 POS2=22
2600 RETURN
2610 POS1=19
2620 POS2=7
2630 RETURN
2640 POS1=19
2650 POS2=10
2660 RETURN
2670 POS1=19
2680 POS2=13
2690 RETURN
2700 POS1=19
2710 POS2=16
2720 RETURN
2730 POS1=19
2740 POS2=19
2750 RETURN
2760 POS1=19
2770 POS2=22
2780 RETURN
2790 IF Q=8 THEN 2820
2800 P=P-100
2810 GOTO 980
2820 IF R>=27 THEN 2860
2830 IF R<=6 THEN 2860
2840 P=P-100
2850 GOTO 980
```



```
2860 CALL VCHAR(Q-1,R,150,2)
2870 Q=22
2880 CALL SOUND(-100,1000,0,1001,0,1002,0)
2890 CALL HCHAR(Q-1,R,112)
2900 CALL HCHAR(Q,R,113)
2910 P=F-1
2920 GOTO 980
2930 CALL SCREEN(16)
2940 CALL CLEAR
2950 PRINT "*****************************"
2960 PRINT :"**********  HELP  **********"
2970 PRINT :"*****************************":::::"IDEE UND AUSFUEHRUNG VON
  ROBERT-JAN MILLEKER"
2980 PRINT :"TEXT UNTER MITARBEIT VON    ALEXANDER STEIN"
2990 FOR RET=1 TO 1500 STEP 15
3000 CALL SOUND(-1000,110+RET,0,5000-RET,0,110,0)
3010 NEXT RET
3020 CALL CLEAR
3030 INPUT "GEBEN SIE BITTE              IHREN NAMEN EIN              " NAME$
3040 CALL CLEAR
3050 CALL CLEAR
3060 PRINT "BENOETIGEN SIE DIE SPIEL-   REGELN? /J/N/"
3070 CALL KEY(0,K,S)
3080 CALL SOUND(-1000,110,0,111,0,110,0)
3090 IF K=74 THEN 3130
3100 IF K=78 THEN 3120
3110 GOTO 3070
3120 RETURN
3130 CALL CLEAR
3140 PRINT "SIE BEFINDEN SICH IN EINER  COMPUTERZENTRALE"
3150 PRINT "SIE MUESSEN DIE AUFLEUCHTEN-DEN LAMPEN VERSUCHEN AUS-   ZULOESCHEN.D
IESES KOENNEN   SIE"
3160 PRINT "MITTELS DER TASTE ""E"" BE-   WIRKEN."
3170 PRINT "DAS MAENNCHEN BEWEGEN SIE   MIT DEN TASTEN              ""/E/S/D/X/"
"."
3180 PRINT "IN DIE OBERE ETAGE KOENNEN  SIE NUR GELANGEN,WENN SIE ANDEN SEITLICH
EN STRICHEN STE-HEN."
3190 PRINT "SE_BIGES GILT,WENN SIE IN   DIE UNTERE ETAGE GELANGEN   WOLLEN."
3200 PRINT "                        >>"
3210 CALL KEY(3,K,S)
3220 CALL SOUND(-1000,110,0)
3230 CALL SOUND(-1000,5000,0)
3240 IF S=0 THEN 3210
3250 PRINT "UNTERWEGS WERDEN SIE VON    ROBOTERN BZW.VON RAUTEN     "
3260 PRINT "BEHINDERT.MIT SPACE BRINGEN SIE DIESE ZUM VERSCHWINDEN."
3270 PRINT :"DRUECKEN SIE EINE TASTE "
3280 CALL KEY(0,K,S)
3290 CALL SOUND(-1000,566,0,566,0,566,0)
3300 IF S=0 THEN 3280
3310 RETURN
3320 CALL CLEAR
3330 END
3340 DATA 330,330,330,300,330,299,301,300,349,349,349,300,392,392,392,300,392,29
2,392,300,349,349,349,300,330
3350 DATA 329,331,300,294,295,293,300,262,263,251,300,262,263,261,300,294,295,29
3,300,330,331,329,300
3360 DATA 330,329,331,550,294,295,293,100,294,295,293,580,330,329,331,300,330,32
9,331,300,349,348,350,300
3370 DATA 392,393,391,300,392,393,391,300,349,350,348,300,330,329,331,300,294,29
5,293,300,262,263,261,300
3380 DATA 262,263,261,300,294,295,293,300,330,329,331,300,294,295,293,470,262,26
3,261,100,262,263,261,520
3390 DATA 294,295,293,300,294,295,293,300,330,329,331,300,262,263,261,300,294,29
5,293,300,330,331,329,100
3400 DATA 349,348,350,100,330,329,331,300,262,263,261,300,294,293,295,300,330,33
1,329,100,349,348,350,100
3410 DATA 330,331,329,300,294,295,293,300,262,263,261,300,294,295,293,300,196,19
7,195,470,330,331,329,300
3420 DATA 330,329,331,300,349,348,350,300,392,393,391,300,392,393,391,300,349,34
8,350,300,330,329,331,300
3430 DATA 294,295,293,300,262,263,261,300,262,263,261,300,294,295,293,300,330,32
9,331,300,294,295,293,470
3440 DATA 262,263,261,100,262,263,261,470
3450 IF NOW>HIGH THEN 3470
3460 RETURN
3470 HIGH=NOW
```

3D-Defence

mit Extended Basic

Am Anfang des Spieles erscheint das Titelbild, mit Riesenbuchstaben und zweifarbig (Riesenbuchstaben in Zeilen: 93Ø-96Ø). Nach dem Drücken einer beliebigen Taste bzw. der Feuertaste oder dem Steuerhebel am Joystick Nr. 1, beginnt das eigentliche Spiel.

Die oberen Amaturen im Raumschiff zeigen High-Score und aktuelle Punktezahl an, die unteren sind für den Treibstoff verantwortlich (FUEL). Ihr Kreuzer benötigt eine gewisse Energie für den Flug, ein Schuß mit dem Laser (Feuertaste am Joystick Nr. 1) verbraucht zusätzliche Energie.
Bei Ihrem routinemäßigen Flug tauchen plötzlich einzeln feindliche Kreaturen oder Ufos auf, um Ihren Raumkreuzer zu rammen. Dies gelingt dem Gegner nur, wenn er nahe genug an Sie herankommt (Kollision mit den unteren Amaturen). Also versuchen Sie, den Feind ins Fadenkreuz zu bekommen. Steuern können Sie den Gegner mit Joystick Nr. 1 (Feind bewegt sich aber entgegengesetzt der Richtung, in die der Steuerhebel gedrückt wurde).
Bei einem Schuß läßt sich der Gegner nicht steuern. Ist nun der Treibstoff aufgebraucht, oder wurde man vom feindlichen Objekt gerammt, erscheint das Endbild mit High-Score und Punktezahl des soeben beendeten Spieles, sowie einer Aufforderung: "Press REDO or BACK". Mit REDO oder BACK wird ein neues Spiel eingeleitet. REDO überspringt das Titelbild.

Organisation des Programmes:

13Ø-18Ø, 26Ø, 29Ø-3ØØ, 93Ø-96Ø Colorbestimmungen und Zeichendefinitionen.
19Ø-25Ø, 27Ø-28Ø Bildschirmaufbau für Hauptteil.
33Ø Bewegt die Sterne
34Ø-39Ø Zeichen und Definitionen für Angreifer auf CHR$(36)
42Ø-43Ø Zeichen für Treibstofftank
13Ø, 44Ø, 92Ø, 94Ø Definition der Variablen
5ØØ-59Ø Setzen der Sterne, Steuerung mit Joystick Nr. 1, Treibstoffroutine
6ØØ-72Ø Schuß und Schußauswertung
73Ø-78Ø Explosion des eigenen Schiffes bei Kollision.
79Ø-87Ø Endbild
91Ø-1Ø4Ø Titelbild
93Ø-96Ø Riesenbuchstaben

Zur Liste mit den Kürzeln:

Der Computer gibt die freidefinierbaren Grafikzeichen (CHR$(128)-CHR$(159)) hinter einem REM-Befehl (bzw. in "TI EXTENDED BASIC" hinter einem !), nicht als die selben wieder, sondern in Form von Statements.
So hat jedes Grafikzeichen seine eigene Funktion (läßt sich über Tastatur mit "CTRL-Taste" und Zeichen abrufen).
Die Eingabe passiert also wie folgt:
Programmzeile
REM (oder in "TI EXTENDED BASIC" !)
Grafikzeichen (CHR$(128)-CHR$(159))
löschen des REM (!)
Beispiel: 1ØØ unterstrichenes U ergibt: 1/ØØ REM RANDOMIZE ergibt 1ØØ RANDOMIZE
P.S. Beim Programm "3D-Defence" handelt es sich in den Zeilen 97Ø und 98Ø bei "_" um CHR$(95), bei Zeilen 19Ø und 24Ø um Grafikzeichen (z. B. unterstrichenes H= CTRL H)

```
100 REM  (C) CHRISTIAN KRAL                    3D-DEFENCE
110 GOTO 910
120 CALL CLEAR :: CALL MAGNIFY(3)
130 J1=0 :: CALL COLOR(1,1,1):: RANDOMIZE
140 CALL CHAR(128,"FFFFFFFFFFFFFFFF",129,"FFFCF0E0C0C08080",130,"8080C0C0E0F0FCF
F",131,"FF3F0F0703030181",132,"8119030307ØF3FFF")
150 CALL CHAR(133,"C381000000000081C3",134,"FFFF7F7F3F3F1F1F",135,"0F0F07070303010
1",136,"8080C0C0E0E0F0F0",137,"F8F8FCFCFEFEFFFF")
160 CALL CHAR(138,"0101030307070F0F",139,"1F1F3F3F7F7FFFFF",140,"FFFFFEFEFCFCF8F
8",141,"F0F0E0E0C0C08080")
170 FOR I=40 TO 47 :: CALL CHAR(I,RPT$("F",16)):: NEXT I :: CALL COLOR(9,4,1,2,7
,1,13,11,1,14,1,1)
180 FOR I=3 TO 8 :: CALL COLOR(I,1,11):: NEXT I
190 DISPLAY AT(1,1):"HI:00000PUNKTE:00000
"
200 DISPLAY AT(11,1):"
210 DISPLAY AT(13,1):"
                      "
220 DISPLAY AT(16,1):"
                     "
230 DISPLAY AT(19,1):"
                    "
240 DISPLAY AT(21,1):"F
UEL( )*+,-./"
250 DISPLAY AT(2,10-LEN(STR$(H))):SIZE(LEN(STR$(H))):STR$(H)
260 CALL CHAR(96,"0101010101010101",97,"8080808080808080",98,"0000000000000000FF",
99,"FF")
270 CALL VCHAR(6,16,96,5):: CALL VCHAR(6,17,97,5) :: CALL VCHAR(13,16,96,5):: CAL
L VCHAR(13,17,97,5)
280 CALL HCHAR(11,6,98,9):: CALL HCHAR(12,6,99,9) :: CALL HCHAR(11,19,98,9):: CAL
L HCHAR(12,19,99,9)
290 CALL CHAR(92,"0000000000001818"&RPT$("0",48),124,"0103060C183060C0"&RPT$("0'
,43))
300 CALL CHAR(60,"C06030180C060301"&RPT$("0",48))
310 GOSUB 330
320 GOTO 410
330 FOR I=4 TO 18 :: CALL SPRITE(#I,92,INT(RND*14+3),83,125,INT(RND*10-5),INT(RN
D*10-5)):: NEXT I
340 F$(1)="000000370F1A3F1F0009020100000000000000F0F8ACFEFC00285040"
350 F$(2)="00000C3601071C2F494A020419000000000018304ØF09CFA49A9A0904C"
360 F$(3)="0000003103060F17A8C2E000000000000000000080D060F0E8852307"
370 F$(4)="0000003111F0103046C936C0000000000000000080F880C02036C936"
380 F$(5)="0000003602030601F0B0A03000000000000000030207ØB0F8E82860"
390 CALL CHAR(36,F$(INT(RND*5+1)))
```

```
400 RETURN
410 RESTORE 430
420 FOR I=1 TO 8 :: READ DATA$ :: CH$(I)=RPT$(DATA$,3):: NEXT I
430 DATA FE,FC,F8,F0,E0,C0,80,00
440 TR=48 :: HC=1-1/8 :: SV=3 :: P=0
450 GOSUB 470
460 GOTO 500
470 ZY=INT(RND*4+1):: ZX=INT(RND*8-4):: IF ZX=0 THEN 470
480 CALL SPRITE(#1,36,INT(RND*14+3),INT(RND*60+20),INT(RND*256+1),ZY,ZX)
490 RETURN
500 SV=SV+1 :: IF SV=19 THEN SV=4
510 CALL LOCATE(#SV,83,125,#SV+4,83,125)
520 CALL SOUND(-4000,-7,17)
530 CALL JOYST(1,X,Y):: CALL MOTION(#1,ZY+Y*1.25,ZX-X*1.25) :: CALL POSITION(#1,Y
Z,XZ):: IF YZ<25 THEN CALL LOCATE(#1,30,XZ)
540 IF YZ>145 AND YZ<185 THEN 730 ELSE IF YZ<256 AND YZ<25 THEN CALL LOCATE(#1,3
0,XZ)
550 GOSUB 570
560 GOTO 600
570 TR=TR-1/64 :: RT=INT(TR):: IF TR=40-1/64 THEN 580 ELSE HC=HC-1/8 :: IF HC=9-
1/8 THEN HC=7/8 :: GOTO 570
580 CALL CHAR(RT,CH$(INT(HC)))
590 RETURN
600 CALL KEY(1,K,S):: IF K<>18 THEN 500
610 CALL MOTION(#1,ZY,ZX):: FOR I=78 TO 1 STEP -4 :: CALL SPRITE(#2,124,6,I+83,1
25-I,#3,60,6,I+83,125+I):: NEXT I
620 CALL MOTION(#1,0,0):: CALL SOUND(-4000,-7,17):: CALL COLOR(#2,2,#3,2)
630 CALL COINC(#2,#1,8,C1):: CALL COINC(#3,#1,8,C2):: IF C1=-1 OR C2=-1 THEN CAL
L DELSPRITE(ALL)ELSE CALL DELSPRITE(#2,#3):: GOTO 1040
640 FOR I=1 TO 20 STEP 2
650 CALL SOUND(-800,-6,I,111,I,112,I,113,I):: CALL SPRITE(#1,92,16,INT(RND*4+84)
,INT(RND*4+123),INT(RND*4-2),INT(RND*4-2))
660 NEXT I
670 FOR I=1 TO 20 :: GOSUB 570 :: NEXT I :: CALL DELSPRITE(ALL)
680 P=P+25 :: DISPLAY AT(2,28-LEN(STR$(P)))SIZE(LEN(STR$(P))):STR$(P)
690 IF NOT H>=P THEN DISPLAY AT(2,10-LEN(STR$(P)))SIZE(LEN(STR$(P))):STR$(P):: H
=P
700 GOSUB 330
710 GOSUB 470
720 GOTO 500
730 CALL DELSPRITE(#1)
740 FOR T=1 TO 5 :: CALL SOUND(-4000,-6,0,1400,19,221,4,339,5):: CALL SCREEN(7):
: CALL COLOR(13,4,1,14,4,1,9,8,1):: CALL SCREEN(5)
750 CALL COLOR(13,9,1,14,9,1,9,16,1):: CALL SCREEN(12):: CALL COLOR(13,5,1,14,5,
1,9,14,1)
760 CALL SCREEN(3):: CALL COLOR(13,14,1,14,14,1,9,10,1):: CALL SCREEN(7):: CALL
SCREEN(8) :: CALL SCREEN(2):: CALL COLOR(13,12,1,14,12,1)
770 CALL COLOR(9,13,1):: CALL SCREEN(13):: CALL SCREEN(7):: CALL SCREEN(8):: CAL
L SCREEN(5):: CALL SCREEN(2):: NEXT T
780 FOR I=0 TO 25 STEP 2 :: CALL SOUND(-200,-6,I,1400,I+4,221,I+1,339,I):: NEXT
I :: J1=1
790 CALL DELSPRITE(ALL):: CALL CLEAR :: FOR I=0 TO 8 :: CALL COLOR(I,5,8):: NEXT
 I
800 DISPLAY AT(1,1):"HI:00000" :: DISPLAY AT(1,9-LEN(STR$(P)))SIZE(5):STR$(P)
810 DISPLAY AT(1,16):"PUNKTE:00000" :: DISPLAY AT(1,28-LEN(STR$(H)))SIZE(5):STR$
(H)
820 IF J1=1 THEN DISPLAY AT(12,1):"SIE   WURDEN   VOM   FEIND":"V E R N I C H
T E T ! !" ELSE 840
830 GOTO 850
840 DISPLAY AT(12,1):"EIN  G U T E S  SPIEL ! ! !":"   NUR   WEITER   SO "
850 DISPLAY AT(7,1)BEEP:" ^ ^ G A M E   O V E R ^ ^" :: DISPLAY AT(22,1):""
860 FOR TI=1 TO 75 :: NEXT TI :: DISPLAY AT(7,1):"" :: DISPLAY AT(22,1):"PRESS
R E D O  OR  B A C K"
870 FOR TI=1 TO 70 :: NEXT TI :: CALL KEY(0,K,S):: IF S=0 THEN 850 ELSE IF K=6 T
HEN 120 ELSE IF K=15 THEN 910 ELSE CALL CLEAR :: STOP
880 CALL MOTION(#1,0,0)
890 CALL SOUND(300,220,1,440,2,880,3):: CALL SOUND(400,330,1,660,2,990,3):: CALL
 SOUND(500,220,3,440,2,880,1)
900 CALL SOUND(1,40000,30):: GOTO 790
910 CALL SCREEN(2):: CALL CLEAR
920 C1$="" :: CALL COLOR(1,5,1,3,8,1):: FOR I=5 TO 8 :: CALL COLOR(I,8,1,I+4,5,1
):: NEXT I
930 CALL CHARPAT(51,I$):: FOR T=1 TO 15 STEP 2 :: C1$=C1$&SEG$(I$,T,2)&SEG$(I$,T
,2):: NEXT T :: CALL CHAR(51,SEG$(C1$,1,16),35,SEG$(C1$,17,16))
940 C1$=""
950 FOR I=65 TO 90 :: CALL CHARPAT(I,I$):: FOR T=1 TO 15 STEP 2 :: C1$=C1$&SEG$(
I$,T,2)&SEG$(I$,T,2):: NEXT T :: CALL CHAR(I,SEG$(C1$,1,16))
960 CALL CHAR(I+32,SEG$(C1$,17,16)):: C1$="" :: NEXT I
970 DISPLAY AT(13,9):'3D_DEFENCE" :: DISPLAY AT(14,9):"#d defence"
980 DISPLAY AT(23,1):'READY_PRESS ANY KEY TO BEGINready press any key to begin"
990 CALL KEY(0,K,S):: IF S=0 THEN 1000 ELSE 1020
1000 CALL KEY(1,K,S):: IF K<>18 THEN 1010 ELSE 1030
1010 CALL JOYST(1,X,Y):: IF X+Y=0 THEN 990
1020 CALL SOUND(200,333,3,444,4,555,5):: CALL SOUND(200,111,1,333,3,555,5):: CAL
L SOUND(400,222,2,555,5,888,8):: CALL CLEAR
1030 CALL CHARSET :: GOTO 120
1040 FOR I=1 TO 20 :: GOSUB 570 :: NEXT I :: GOTO 500
```

SPACE CHASE

Space-Chase ist ein Weltraumspielprogramm, für daß das Extended-Basic-Modul benötigt wird.

Zu Beginn hat der Spieler drei Raumschiffe (eines davon ist auf dem Bildschirm sichtbar). Gleichzeitig fliegen rote, weiße und gelbe (grüne) Schiffe einer feindlichen Macht durch das All.
Die Aufgabe des Spielers besteht darin, das gelbe Schiff zu rammen. Gelingt die Kollision nicht in kurzer Zeit, so verwandelt sich das gelbe Schiff in ein grünes. In diesem Fall muß versucht werden, das grüne Schiff zu rammen, dies bringt aber weniger Punkte ein, als bei dem gelben.
Wurde ein gelbes (grünes) Schiff gerammt, dann verschwindet es und eines der bisher weißen Schiffe verwandelt sich in ein gelbes.
Eine Kollision mit einem weißen Schiff hat keinen Effekt.
Die Kollision mit roten Schiffen ist auf jeden Fall zu vermeiden, da dabei das eigene zerstört wird.
Ausnahme ist, wenn außer einem roten Schiff keine weiteren gegnerischen Schiffe auf dem Bildschirm sind. Dann muß man das rote Schiff fangen, was aber schwierig ist, da es seine Richtung und Geschwindigkeit schnell ändert.
Hat man auch dieses rote Schiff zerstört, so ist eine Runde beendet. In den nächsten drei Runden nimmt jeweils die Geschwindigkeit der gegnerischen Raumschiffe zu, während die des eigenen Schiffes durch den gesamten Spielverlauf konstant bleibt.
In den vier folgenden Runden spielt sich jeweils das gleiche ab, wie in den vier ersten Runden, nur daß jetzt das jeweils zu vernichtende Schiff getarnt ist. D. h. es ist nicht mehr gelb (grün) sondern genauso weiß, wie die anderen Schiffe. Auch hier erhält man wieder eine viel höhere Punktzahl, wenn man das Schiff schnell findet und vernichtet. Die 8 letzten Runden sind wiederum identisch mit 8 ersten, nur daß jetzt nicht nur ein rotes, sondern zwei rote Schiffe die Gegend unsicher machen.
Wird man von einem roten Schiff getroffen, so muß man die Runde wiederholen und der Tank wird nicht aufgefüllt.
Ist der Tank leer oder hat man keine Schiffe mehr, so ist das Spiel aus. Hat man eine Runde erfolgreich absolviert, dann wird der Tank wieder aufgefüllt. Erreicht man eine Punktzahl von 20000, so erhält man ein Zusatzschiff. Die Punktzahl, die man für die Vernichtung eines feindlichen Schiffes erhält, hängt auch von der Schwierigkeit der Spielrunde ab. Permanent auf dem Bildschirm sichtbar sind die aktuelle Punktzahl, der High-Score, das zweitbeste Ergebnis, sowie die Tankauffüllung und in Form von kleinen Kästchen, die die Form des Schiffes haben ist auch die Anzahl der Schiffe, die man noch in Reserve hat, ständig sichtbar.
Das Schiff kann mittels der Tasten E,S,D,X gesteuert werden.
Drückt man die Taste 1, (Hyperspace) so wird das Schiff an eine nicht vorhersehbare Stelle "geschleudert", die auch außerhalb des sichtbaren Bildschirmbereiches liegen kann.
Betätigt man irgendeine andere Taste des linken Tastenfeldes, so hält das Raumschiff an.

```
70 !SPACE CHASE
80 !COPYRIGHT 1983 BY
90 !F. SCHWELLINGER
100 DIM AA(20),BB(20)
110 AA(5),BB(2)=-20
120 AA(0),BB(3)=20
130 Z,HI=0
140 CALL CLEAR
150 FOR K=5 TO 10
160 CALL COLOR(K,2,14)
170 NEXT K
180 FOR W=3 TO 4
190 CALL COLOR(W,14,2)
200 NEXT W
210 DISPLAY AT(1,5):"SCORE  0"
220 DISPLAY AT(4,5):"HIGH SCORE  0"
230 DISPLAY AT(6,13):"2nd 0"
240 DEF DIR(I)=(RND*SP+I)*SGN(.5-RND)
250 CALL SCREEN(2)
260 RANDOMIZE
270 CALL MAGNIFY(2)
280 CALL CHAR(99,"FF7E3C10083C7EFF")
290 CALL CHAR(96,"10183C7FFE3C1808")
300 DISPLAY AT(12,10):"PLAY SPACE CHASE"
310 DISPLAY AT(15,5):"PRESS ANY KEY TO BEGIN"
320 CALL KEY(0,XXX,F)
330 HH=0 :: CO=500
340 IF F=0 THEN 320
350 DISPLAY AT(12,10):RPT$(" ",16)
360 DISPLAY AT(15,5):RPT$(" ",22)
370 SHIPS=3 :: CALL SOUND(1500,440,0)
380 DISPLAY AT(1,12):0
390 TB=0
```

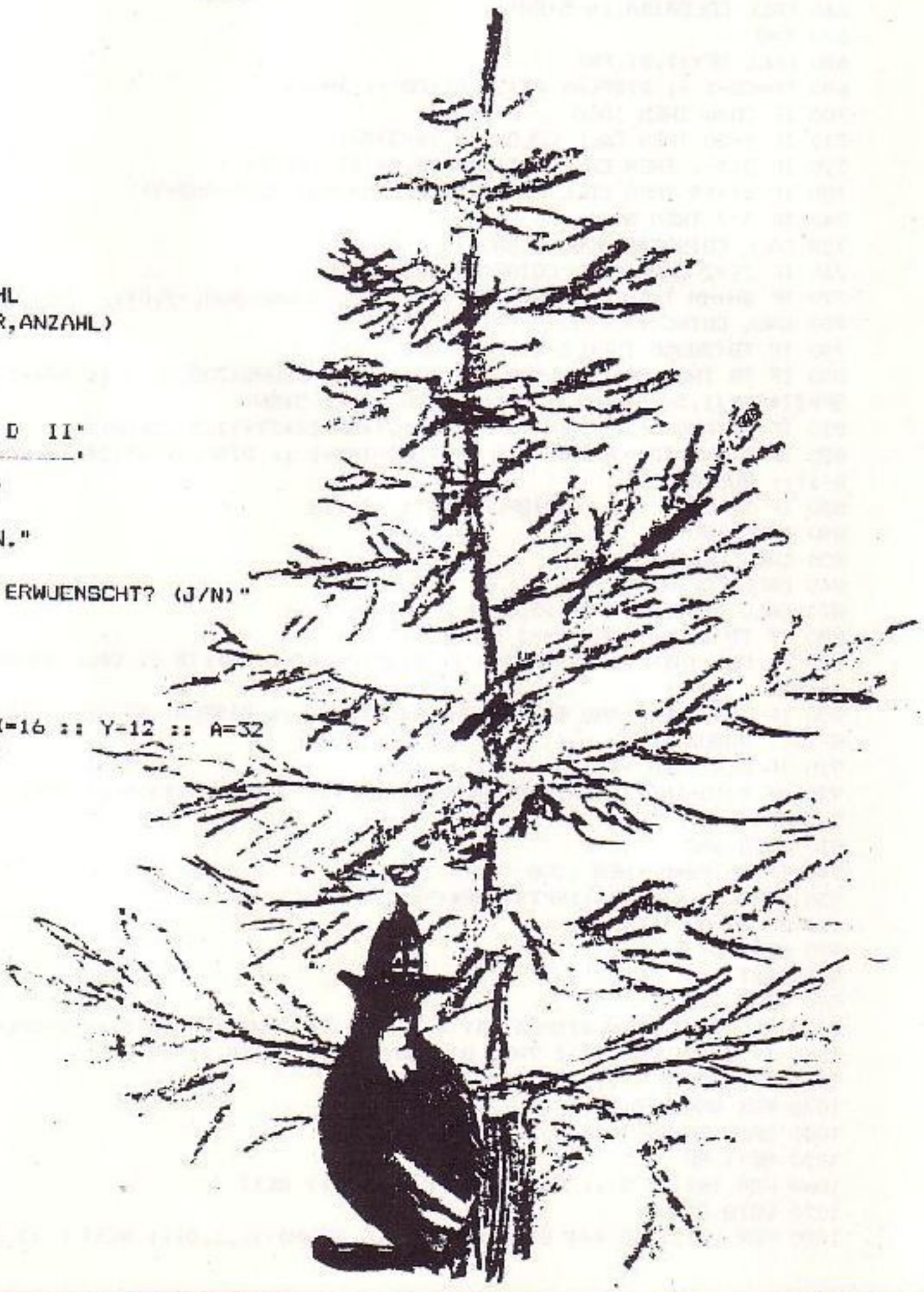
```
400 CALL HCHAR(24,3,96,2)
410 DISPLAY AT(22,1):"TANK:  500"
420 FOR ZT=1 TO 2
430 FOR SH=1 TO 0 STEP -1
440 FOR SP=10 TO 25 STEP 5
450 FOR SZ=1 TO 7
460 CALL SPRITE(#SZ,99,16,1,1,DIR(1),DIR(1))
470 NEXT SZ
480 FOR TOS=8 TO ZT+7
490 CALL SPRITE(#TOS,99,9,1,1,DIR(5),DIR(5))
500 NEXT TOS
510 CALL SPRITE(#28,96,6,81,128,0,0)
520 FOR I=1 TO 5
530 FOR X=8 TO ZT+7
540 CALL COLOR(#X,9)
550 NEXT X
560 CALL SOUND(300,2500,0)
570 FOR X=8 TO ZT+7
580 CALL COLOR(#X,1)
590 NEXT X
600 CALL SOUND(600,40000,30)
610 FOR X=8 TO ZT+7
620 CALL COLOR(#X,9)
630 NEXT X
640 NEXT I
650 A=1
660 CALL COLOR(#A,16-5*SH)
670 X=0
680 CALL KEY(1,DI,PR)
690 CO=CO-1 :: DISPLAY AT(22,8):CO :: X=X+1
700 IF CO=0 THEN 1000
710 IF X=20 THEN CALL COLOR(#A,16-3*SH)
720 IF DI>-1 THEN CALL MOTION(#28,AA(DI),BB(DI))
730 IF DI=19 THEN CALL LOCATE(#28,255*RND+1,255*RND+1)
740 IF A>7 THEN 850
750 CALL COINC(#8,#28,10,GH)
760 IF ZT=2 THEN CALL COINC(#28,#9,10,HH)
770 IF GH+HH THEN SHIPS=SHIPS-1 :: CALL SOUND(800,-7,0):: GOTO 940
780 CALL COINC(#A,#28,15,TR)
790 IF TB<20000 THEN E=-1 ELSE E=0
800 IF TR THEN CALL DELSPRITE(#A):: CALL SOUND(700,-2,0):: A=A+1 :: TB=INT(TB+3*
SP*ZT*(1*(1.5-.5*SH)):: CALL COLOR(#A,16-5*SH)
810 IF TR AND X<20 THEN TB=INT(TB+27*SP*ZT*ZT*(1.5-.5*SH))
820 IF E AND TB>=20000 THEN SHIPS=SHIPS+1 :: DISPLAY AT(24,1):RPT$(CHR$(96),SHIP
S-1):: GOSUB 1080
830 IF TR THEN X=0 :: DISPLAY AT(1,12):TB
840 GOTO 680
850 CALL DELSPRITE(#9)
860 CALL COLOR(#8,9)
870 CALL COINC(#28,#8,15,DES)
880 IF TB<20000 THEN E=-1 ELSE E=0
890 IF DES THEN TB=TB+SP*200 :: DISPLAY AT(1,12):TB :: CALL SOUND(4000,110,0)::
CO=500
900 IF TB>=20000 AND E THEN SHIPS=SHIPS+1 :: DISPLAY AT(24,1):RPT$(CHR$(96),SHIP
S-1):: GOSUB 1080
910 IF DES THEN 940
920 IF X/10-INT(X/10)THEN CALL MOTION(#8,(30*RND+1)*SGN(.5-RND),(30*RND+1)*SGN(.
5-RND))
930 GOTO 680
940 IF SHIPS=0 THEN 1000
950 DISPLAY AT(24,1):RPT$(CHR$(96),SHIPS-1)
960 IF GH+HH THEN 450
970 NEXT SP
980 NEXT SH
990 NEXT ZT
1000 IF TB>HI THEN DISPLAY AT(4,17):TB :: Z=HI :: HI=TB :: DISPLAY AT(6,17):Z
1010 IF TB<HI AND TB>Z THEN DISPLAY AT(6,17):TB :: Z=TB
1020 DISPLAY AT(12,12):"GAME OVER"
1030 FOR NM=1 TO 50
1040 CALL SOUND(10,5000,0)
1050 NEXT NM
1060 FOR I=1 TO 9 :: CALL DELSPRITE(#I):: NEXT I
1070 GOTO 300
1080 FOR L=111 TO 444 STEP 10 :: CALL SOUND(10,L,0):: NEXT L :: RETURN
```

Pilzjagd

Extended Basic

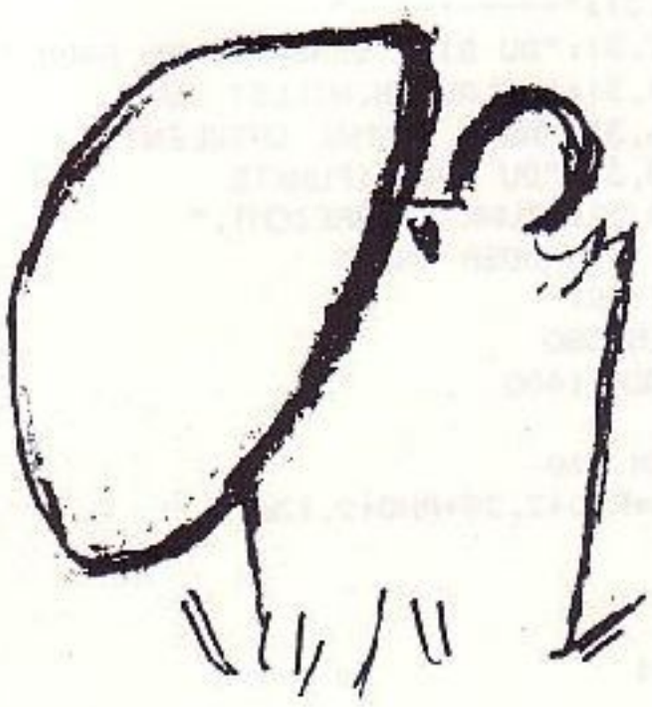
**Das Programm "Pilzjagd" läuft auf dem Ti mit Extended Basic.
Sie müssen viele Pilze sammeln, doch Bäume, die im Weg stehen, behindern Sie dabei. Geister laufen Ihnen nach und versuchen, das Pilzesammeln zu erschweren. Färben sich die ursprünglich blauen Geister rot, können sie gefressen werden. Gesteuert wird über die Cursortasten. Es ist auch möglich, Sprünge über Bäume und Geister auszuführen. Dabei werden allerdings Punkte abgezogen. Beendet ist das Spiel, wenn das Punktekonto negativ wird, oder wenn man von einem Geist eingeholt wird. Mit steigender Punktzahl werden auch die Geister schneller. Das Programm zeichnet sich durch gelungene Grafik aus.**

```
100 !************************
110 !*                      *
120 !*     PILZJAGD II      *
130 !*     ----------       *
140 !*                      *
150 !*                      *
160 !*    COPYRIGHT BY      *
170 !*                      *
180 !*  MATTHIAS NAUMANN    *
190 !*                      *
200 !*                      *
210 !* ALL RIGHTS RESERVED *
220 !* ------------------- *
230 !*                      *
240 !*     11904 BYTES      *
250 !*                      *
260 !*     OCTOBER 1983     *
270 !*                      *
280 !************************
290 !
300 !******************
310 !***            ***
320 !*** DEFINITION ***
330 !***            ***
340 !******************
350 !
360 CALL CLEAR
370 FOR I=1 TO 19
380 READ CHARACTER,CODE$
390 CALL CHAR(CHARACTER,CODE$)
400 NEXT I
410 CALL SCREEN(2)
420 FOR I=1 TO 34
430 READ ZEILE,SPALTE,CHARACTER,ANZAHL
440 CALL HCHAR(ZEILE,SPALTE,CHARACTER,ANZAHL)
450 NEXT I
460 CALL COLOR(10,9,1,11,9,1,12,9,1)
470 CALL SCREEN(8)
480 DISPLAY AT(19,5):" P I L Z J A G D  II"
490 DISPLAY AT(20,5):" ------------------"
500 CALL CHAR(128,"3C4299A1A199423C")
510 CALL HCHAR(22,6,128)
520 DISPLAY AT(22,8):"OCT. '83 BY M.N."
530 GOSUB 3260
540 DISPLAY BEEP AT(24,2):"ANLEITUNG ERWUENSCHT? (J/N)"
550 CALL KEY(0,K,S)
560 IF K=74 THEN 3380
570 IF K<>78 THEN 550
580 CALL CLEAR
590 PILZ=0 :: BAUM=0 :: PUNKTE=0 :: X=16 :: Y=12 :: A=32
600 CALL CHAR(128,"3C7FFF7E7C181818")
610 CALL COLOR(14,9,1,13,3,1)
620 CALL SCREEN(15)
630 FOR I=1 TO 30
640 A=A-1
650 FOR J=1 TO 3
660 CALL HCHAR(22*RND+1,A,128,RND*2)
670 NEXT J
680 NEXT I
690 CALL HCHAR(1,1,128,32)
700 CALL VCHAR(1,1,128,23)
710 CALL VCHAR(1,32,128,23)
720 CALL HCHAR(23,1,128,32)
730 CALL HCHAR(Y,X,42)
740 !
750 !******************
760 !***            ***
770 !*** SPIELABLAUF ***
780 !***            ***
790 !******************
```

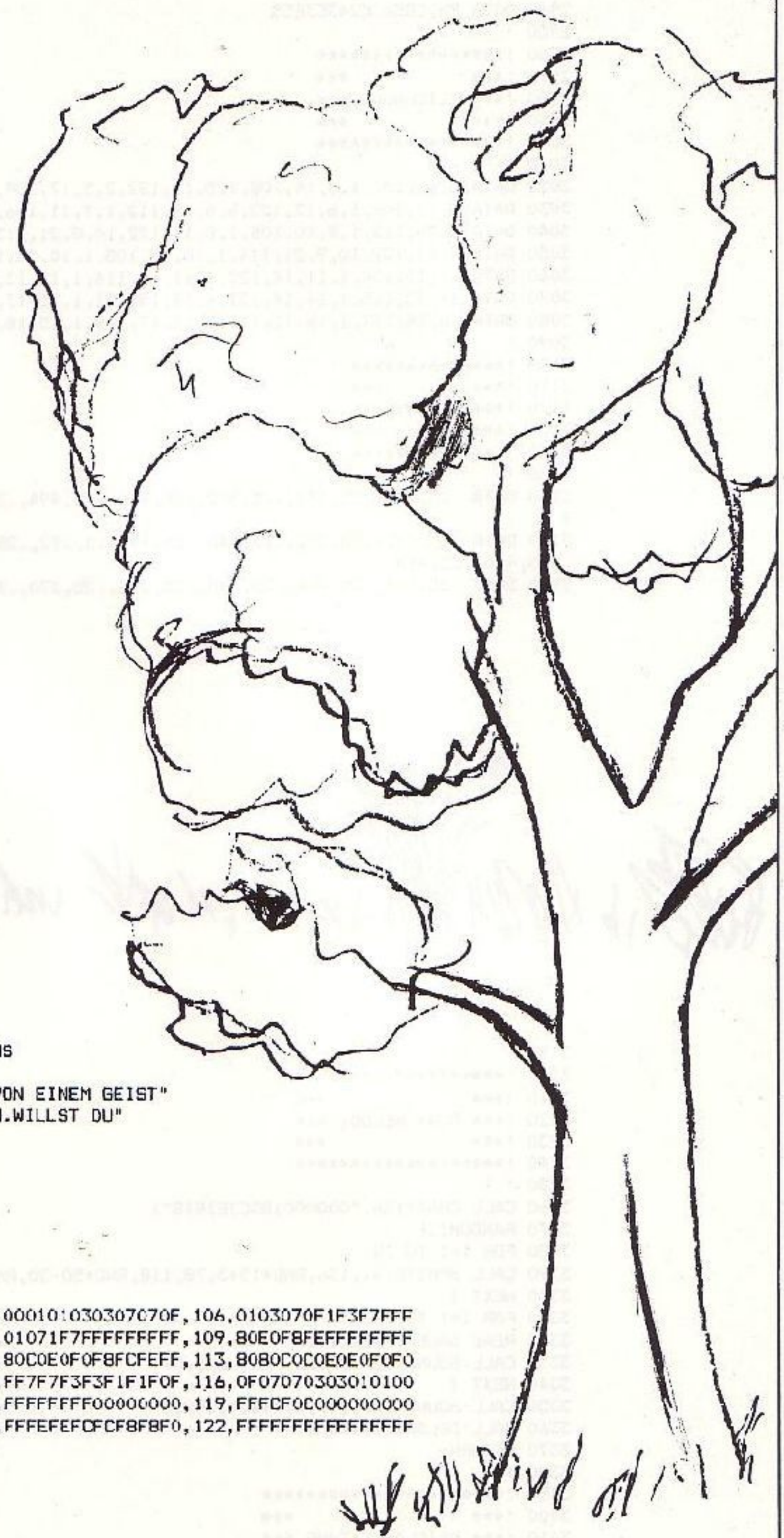


```
800 !
810 BAUM=BAUM+1 :: IF BAUM>100 THEN BAUM=1
820 IF BAUM-50>=0 THEN 1610
830 CALL COLOR(9,10,1):: WERT=1 :: GEIST=0
840 ACHSE1=22*RND+1 :: ACHSE2=30*RND+1
850 CALL GCHAR(ACHSE1,ACHSE2,PILZX)
860 IF PILZX=136 THEN PILZ=PILZ-1
870 CALL HCHAR(ACHSE1,ACHSE2,128)
880 IF X=1 THEN X=X+1
890 IF Y=24 THEN Y=Y-1
900 IF Y=1 THEN Y=Y+1
910 IF X=32 THEN X=X-1
920 GOTO 1440
930 CALL HCHAR(Y,X,42)
940 CALL KEY(1,K,S)
950 IF K=2 THEN X=X-1
960 IF K=3 THEN X=X+1
970 IF K=5 THEN Y=Y-1
980 IF K=0 THEN Y=Y+1
990 IF K=6 THEN 4190
1000 CALL GCHAR(Y,X,A)
1010 GOSUB 1130
1020 IF A=96 AND WERT=1 THEN PUNKTE=PUNKTE+50 :: CALL SOUND(500,330,0,392,0,523,
0)
1030 IF A=128 THEN 1080
1040 IF A=136 THEN PUNKTE=PUNKTE+10 :: PILZ=PILZ-1 :: GOSUB 1480 :: GOTO 1440
1050 Z=.25 :: Z=Z+PUNKTE/1000 :: IF Z>.75 THEN Z=.75
1060 DISPLAY AT(24,1):"PUNKTE:";PUNKTE
1070 GOTO 810
1080 FOR I=1100 TO 200 STEP -100
1090 CALL SOUND(50,I,0)
1100 NEXT I
1110 CALL HCHAR(Y,X,128)
1120 GOTO 1300
1130 IF K=2 THEN 1180
1140 IF K=3 THEN 1210
1150 IF K=5 THEN 1240
1160 IF K=0 THEN 1270
1170 RETURN
1180 CALL HCHAR(Y,X,42)
1190 CALL HCHAR(Y,X+1,32)
1200 RETURN
1210 CALL HCHAR(Y,X,42)
1220 CALL HCHAR(Y,X-1,32)
1230 RETURN
1240 CALL HCHAR(Y,X,42)
1250 CALL HCHAR(Y+1,X,32)
1260 RETURN
1270 CALL HCHAR(Y,X,42)
1280 CALL HCHAR(Y-1,X,32)
1290 RETURN
1300 CALL CLEAR
1310 HS=MAX(PUNKTE,HS)
1320 DISPLAY AT(5,3):"HIGHSCORE:";HS
1330 DISPLAY AT(6,3):"----------"
1340 DISPLAY AT(12,3):"DU BIST GEGEN EINEN BAUM "
1350 DISPLAY AT(14,3):"GELAUFEN.WILLST DU"
1360 DISPLAY AT(16,3):"NOCH EINMAL SPIELEN?"
1370 DISPLAY AT(18,3):"DU HAST";PUNKTE
1380 DISPLAY AT(20,3):"PUNKTE ERREICHT."
1390 PRINT TAB(3):"(J) ODER (N):"
1400 CALL KEY(0,KEY,S)
1410 IF KEY=74 THEN 580
1420 IF KEY<>78 THEN 1400
1430 END
1440 IF PILZ=5 THEN 940
1450 CALL HCHAR(20*RND+2,29*RND+2,136)
1460 PILZ=PILZ+1
1470 GOTO 940
1480 FOR I=1 TO 3
1490 FOR J=-3 TO -1
```

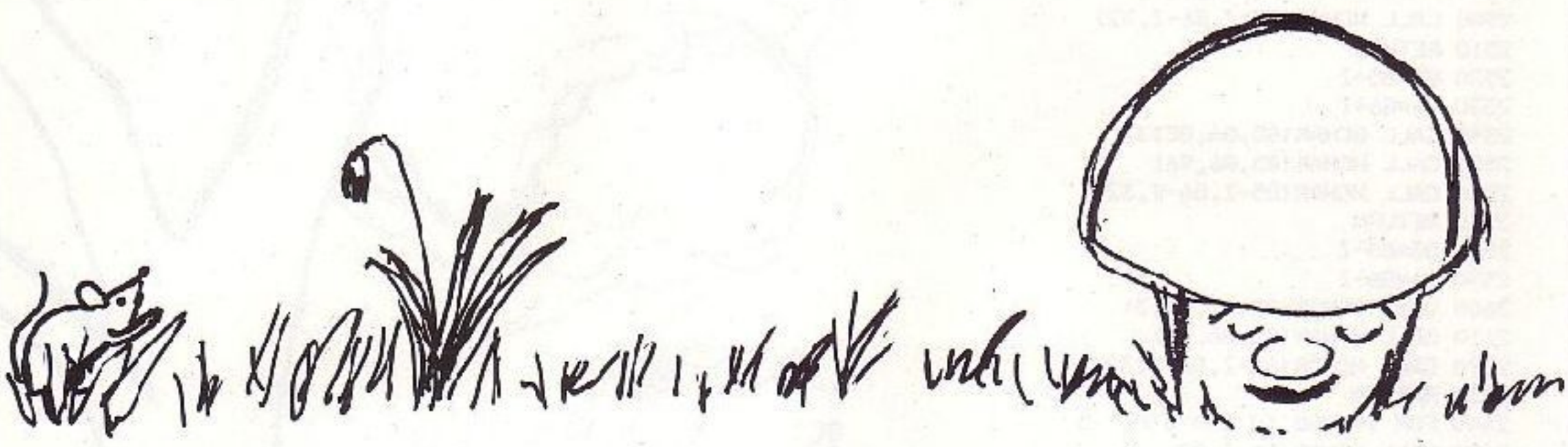
```
1500 CALL SOUND(50,J,0)
1510 NEXT J
1520 NEXT I
1530 RETURN
1540 !
1550 !****************
1560 !***          ***
1570 !*** GEISTER ***
1580 !***          ***
1590 !****************
1600 !
1610 WERT=0 :: CALL COLOR(9,6,1)
1620 GEIST=GEIST+1
1630 IF GEIST>3 THEN 1770
1640 ON GEIST GOTO 1650,1690,1730
1650 G1=20*RND+2
1660 G2=29*RND+2
1670 CALL HCHAR(G1,G2,96)
1680 GOTO 1620
1690 G3=20*RND+2
1700 G4=29*RND+2
1710 CALL HCHAR(G3,G4,96)
1720 GOTO 1620
1730 G5=20*RND+2
1740 G6=29*RND+2
1750 CALL HCHAR(G5,G6,96)
1760 GOTO 1620
1770 IF Y>G1 AND X<G2 THEN GOSUB 1920
1780 IF Y<G1 AND X>G2 THEN GOSUB 1980
1790 IF Y>G1 AND X>G2 THEN GOSUB 2040
1800 IF Y<G1 AND X<G2 THEN GOSUB 2100
1810 IF Y>G3 AND X<G4 THEN GOSUB 2160
1820 IF Y<G3 AND X>G4 THEN GOSUB 2220
1830 IF Y>G3 AND X>G4 THEN GOSUB 2280
1840 IF Y<G3 AND X<G4 THEN GOSUB 2340
1850 IF Y>G5 AND X<G6 THEN GOSUB 2400
1860 IF Y<G5 AND X>G6 THEN GOSUB 2460
1870 IF Y>G5 AND X>G6 THEN GOSUB 2520
1880 IF Y<G5 AND X<G6 THEN GOSUB 2580
1890 IF GEI1=42 OR GEI2=42 OR GEI3=42 THEN 2640
1900 IF GEI1=136 OR GEI2=136 OR GEI3=136 THEN PILZ=PILZ-1
1910 GOTO 850
1920 G1=G1+Z
1930 G2=G2-Z
1940 CALL GCHAR(G1,G2,GEI1)
1950 CALL HCHAR(G1,G2,96)
1960 CALL HCHAR(G1-Z,G2+Z,32)
1970 RETURN
1980 G1=G1-Z
1990 G2=G2+Z
2000 CALL GCHAR(G1,G2,GEI1)
2010 CALL HCHAR(G1,G2,96)
2020 CALL HCHAR(G1+Z,G2-Z,32)
2030 RETURN
2040 G1=G1+Z
2050 G2=G2+Z
2060 CALL GCHAR(G1,G2,GEI1)
2070 CALL HCHAR(G1,G2,96)
2080 CALL HCHAR(G1-Z,G2-Z,32)
2090 RETURN
2100 G1=G1-Z
2110 G2=G2-Z
2120 CALL GCHAR(G1,32,GEI1)
2130 CALL HCHAR(G1,G2,96)
2140 CALL HCHAR(G1+Z,G2+Z,32)
2150 RETURN
2160 G3=G3+Z
2170 G4=G4-Z
2180 CALL GCHAR(G3,G4,GEI2)
2190 CALL HCHAR(G3,G4,96)
2200 CALL HCHAR(G3-Z,G4+Z,32)
2210 RETURN
```



```
2220 G3=G3-Z
2230 G4=G4+Z
2240 CALL GCHAR(G3,G4,GEI2)
2250 CALL HCHAR(G3,G4,96)
2260 CALL HCHAR(G3+Z,G4-Z,32)
2270 RETURN
2280 G3=G3+Z
2290 G4=G4+Z
2300 CALL GCHAR(G3,G4,GEI2)
2310 CALL HCHAR(G3,G4,96)
2320 CALL HCHAR(G3-Z,G4-Z,32)
2330 RETURN
2340 G3=G3-Z
2350 G4=G4-Z
2360 CALL GCHAR(G3,G4,GEI2)
2370 CALL HCHAR(G3,G4,96)
2380 CALL HCHAR(G3+Z,G4+Z,32)
2390 RETURN
2400 G5=G5+Z
2410 G6=G6-Z
2420 CALL GCHAR(G5,G6,GEI3)
2430 CALL HCHAR(G5,G6,96)
2440 CALL HCHAR(G5-Z,G6+Z,32)
2450 RETURN
2460 G5=G5-Z
2470 G6=G6+Z
2480 CALL GCHAR(G5,G6,GEI3)
2490 CALL HCHAR(G5,G6,96)
2500 CALL HCHAR(G5+Z,G6-Z,32)
2510 RETURN
2520 G5=G5+Z
2530 G6=G6+Z
2540 CALL GCHAR(G5,G6,GEI3)
2550 CALL HCHAR(G5,G6,96)
2560 CALL HCHAR(G5-Z,G6-Z,32)
2570 RETURN
2580 G5=G5-Z
2590 G6=G6-Z
2600 CALL GCHAR(G5,G6,GEI3)
2610 CALL HCHAR(G5,G6,96)
2620 CALL HCHAR(G5+Z,G6+Z,32)
2630 RETURN
2640 FOR I=1000 TO 500 STEP -5
2650 CALL SOUND(-1,I,0)
2660 NEXT I
2670 CALL CLEAR
2680 HS=MAX(PUNKTE,HS)
2690 DISPLAY AT(5,3):"HIGHSCORE:";HS
2700 DISPLAY AT(6,3):"----------"
2710 DISPLAY AT(12,3):"DU WURDEST VON EINEM GEIST"
2720 DISPLAY AT(14,3):"AUFGEFRESSEN.WILLST DU"
2730 GOTO 1360
2740 !
2750 !************
2760 !***      ***
2770 !*** PIL2 ***
2780 !***      ***
2790 !************
2800 !
2810 DATA 104,0F1F3F3F7F7FFF,105,00010103030707070F,106,0103070F1F3F7FFF
2820 DATA 107,0000000001071F7F,108,01071F7FFFFFFFFF,109,80E0F8FEFFFFFFFF
2830 DATA 110,00000000B0FCFCFF,112,80C0E0F0F8FCFEFF,113,8080C0C0E0E0F0F0
2840 DATA 114,F8F8FCFCFEFEFFFF,115,FF7F7F3F3F1F1F0F,116,0F0707030301010100
2850 DATA 117,FF3F0F0300000000,118,FFFFFFFF00000000,119,FFFCF0C000000000
2860 DATA 120,F0F0E0E0C0C08080,121,FFFEFEFCFCF8F8F0,122,FFFFFFFFFFFFFFFF
2870 !
2880 !************
2890 !***      ***
2900 !*** GEIST ***
2910 !***      ***
2920 !************
2930 !
```



```
2940 DATA 96,C85A3C243C3E55
2950 !
2960 !*******************
2970 !***               ***
2980 !*** ZEICHNUNG ***
2990 !***               ***
3000 !*******************
3010 !
3020 DATA 5,13,107,1,5,14,108,1,5,15,122,2,5,17,109,1,5,18,110,1
3030 DATA 6,12,106,1,6,13,122,6,6,19,112,1,7,11,106,1,7,12,122,8
3040 DATA 7,20,112,1,8,10,105,1,8,11,122,10,8,21,113,1,9,10,104,1
3050 DATA 9,11,122,10,9,21,114,1,10,13,105,1,10,14,122,4,10,18,113,1
3060 DATA 11,13,104,1,11,14,122,4,11,18,114,1,12,13,122,6,13,13,122,6
3070 DATA 14,13,115,1,14,14,122,4,14,18,121,1,15,13,116,1,15,14,122,4
3080 DATA 16,14,117,1,16,15,118,2,16,17,119,1,15,18,120,1
3090 !
3100 !***************
3110 !***           ***
3120 !*** MELODY ***
3130 !***           ***
3140 !***************
3150 !
3160 DATA .25,294,.25,392,.25,392,.25,440,.25,494,.25,392,.25,494,.,.25,29
4
3170 DATA .25,392,.25,392,.25,440,.25,494,.5,392,.25,370,.25,294,.25,392,.25,392
,.25,440,.25,494
3180 DATA .25,523,.25,494,.25,440,.25,392,.25,370,.25,294,.25,330,.25,370,.5,392
```



```
3190 !
3200 !*********************
3210 !***                 ***
3220 !*** PLAY MELODY ***
3230 !***                 ***
3240 !*********************
3250 !
3260 CALL CHAR(136,"0000001B3C7E1818")
3270 RANDOMIZE
3280 FOR I=1 TO 28
3290 CALL SPRITE(#I,136,RND*13+3,78,118,RND*50-30,RND*50-30)
3300 NEXT I
3310 FOR I=1 TO 29
3320 READ DAUER,FREQUENZ
3330 CALL SOUND(750*DAUER,FREQUENZ,0)
3340 NEXT I
3350 CALL SOUND(500,247,0,294,0,392,0)
3360 CALL DELSPRITE(ALL)
3370 RETURN
3380 !
3390 !************************
3400 !***                  ***
3410 !*** SPIELANLEITUNG ***
3420 !***                  ***
3430 !************************
```

```
3440 !
3450 CALL CLEAR
3460 CALL CHAR(128,"3C7FFF7E7C181818")
3470 CALL SCREEN(5):: FOR I=2 TO 14 :: CALL COLOR(I,16,6):: NEXT I :: CALL COLOR
(1,6,6)
3480 DISPLAY AT(3,2):"DER SINN DES SPIELES LIEGT"
3490 DISPLAY AT(5,2):"DARIN, DASS MAN MIT DER"
3500 DISPLAY AT(7,2):"SPIELERFIGUR (*) MOEGLICHST"
3510 DISPLAY AT(9,2):"VIELE PILZE (";CHR$(136);") SAMMELT."
3520 DISPLAY AT(11,2):"JEDER PILZ IST 10 PUNKTE"
3530 DISPLAY AT(13,2):"WERT."
3540 DISPLAY AT(15,2):"GEGEN DIE BAEUME (";CHR$(128);")"
3550 DISPLAY AT(17,2):"DARF MAN NICHT LAUFEN;"
3560 DISPLAY AT(19,2):"DAS SPIEL IST DANN ZU "
3570 DISPLAY AT(21,2):"ENDE."
3580 DISPLAY AT(23,2):"BITTE EINE TASTE DRUECKEN"
3590 CALL KEY(0,K,S)
3600 IF S=0 THEN 3590 ELSE CALL CLEAR
3610 DISPLAY AT(3,2):"DIE STEUERUNG DER SPIELER-"
3620 DISPLAY AT(5,2):"FIGUR ERFOLGT MIT DEN"
3630 DISPLAY AT(7,2):"PFEILTASTEN:"
3640 DISPLAY AT(9,14):"E"
3650 CALL CHAR(143,"1010202044444400")
3660 DISPLAY AT(11,14):CHR$(143)
3670 DISPLAY AT(12,11):"S <*> D"
3680 DISPLAY AT(13,14):"V"
3690 DISPLAY AT(15,14):"X"
3700 DISPLAY AT(23,2):"BITTE EINE TASTE DRUECKEN"
3710 CALL KEY(0,K,S)
3720 IF S=0 THEN 3710 ELSE CALL CLEAR
3730 DISPLAY AT(3,2):"AUCH VOR DEN GELEGENTLICH"
3740 DISPLAY AT(5,2):"AUFTAUCHENDEN GEISTERN (";CHR$(96);")"
3750 DISPLAY AT(7,2):"MUSS MAN SICH HUETEN,"
3760 DISPLAY AT(9,2):"DENN WENN MAN VON IHNEN"
3770 DISPLAY AT(11,2):"EINGEHOLT WIRD, WIRD MAN"
3780 DISPLAY AT(13,2):"GEFRESSEN. WENN SIE JEDOCH"
3790 DISPLAY AT(15,2):"ROT SIND, SO KANN MAN"
3800 DISPLAY AT(17,2):"SIE SELBST FRESSEN UND ER-"
3810 DISPLAY AT(19,2):"HAELT FUER JEDEN GEIST 50"
3820 DISPLAY AT(21,2):"PUNKTE."
3830 DISPLAY AT(23,2):"BITTE EINE TASTE DRUECKEN"
3840 CALL KEY(0,K,S)
3850 IF S=0 THEN 3840 ELSE CALL CLEAR
3860 DISPLAY AT(3,2):"FERNER MUSS MAN SICH VOR"
3870 DISPLAY AT(5,2):"PLOETZLICH AUFTAUCHENDEN "
3880 DISPLAY AT(7,2):"BAEUMEN HUETEN. DIESE "
3890 DISPLAY AT(9,2):"KOENNEN NUR UMGANGEN "
3900 DISPLAY AT(11,2):"WERDEN. MIT DER TASTE (R)"
3910 DISPLAY AT(13,2):"KANN EIN SPRUNG AUSGEFUEHRT"
3920 DISPLAY AT(15,2):"WERDEN, DER ABER MIT EINEM"
3930 DISPLAY AT(17,2):"RISIKO VERBUNDEN IST."
3940 DISPLAY AT(19,2):"(SPRUNG AUF EINEN BAUM ODER"
3950 DISPLAY AT(21,2):"EINEN GEIST)"
3960 DISPLAY AT(23,2):"BITTE EINE TASTE DRUECKEN"
3970 CALL KEY(0,K,S)
3980 IF S=0 THEN 3970 ELSE CALL CLEAR
3990 DISPLAY AT(3,2):"JEDER SPRUNG KOSTET 20 "
4000 DISPLAY AT(5,2):"PUNKTE. WENN DAS PUNKTE-"
4010 DISPLAY AT(7,2):"KONTO NEGATIV IST, IST DAS"
4020 DISPLAY AT(9,2):"SPIEL ZU ENDE."
4030 DISPLAY AT(11,2):"ZU BEACHTEN IST AUCH,DASS"
4040 DISPLAY AT(13,2):"DIE GEISTER MIT STEIGENDER"
4050 DISPLAY AT(15,2):"PUNKTEZAHL SCHNELLER "
4060 DISPLAY AT(17,2):"WERDEN."
4070 DISPLAY AT(19,5):"VIEL GLUECK"
4080 DISPLAY AT(20,5):"----------"
4090 DISPLAY AT(23,2):"BITTE EINE TASTE DRUECKEN"
4100 CALL KEY(0,K,S):: IF S=0 THEN GOTO 4100 ELSE CALL CLEAR
4110 FOR I=0 TO 14 :: CALL COLOR(I,2,1):: NEXT I :: GOTO 590
4120 !
```

```
4130 !***************
4140 !***           ***
4150 !*** SPRUNG ***
4160 !***           ***
4170 !***************
4180 !
4190 CALL HCHAR(Y,X,32):: Y=20*RND+2 :: X=28*RND+2 :: CALL SOUND(100,-1,0):: CAL
L SOUND(100,-3,0):: CALL GCHAR(Y,X,AB)
4200 PUNKTE=PUNKTE-20 :: IF PUNKTE<0 THEN CALL CLEAR :: GOTO 4260
4210 IF AB=128 THEN 1300
4220 IF AB=96 THEN 2670
4230 CALL HCHAR(Y,X,42)
4240 IF AB=136 THEN 1040
4250 GOTO 1000
4260 HS=MAX(PUNKTE,HS)
4270 DISPLAY AT(5,3):"HIGHSCORE:";HS
4280 DISPLAY AT(6,3):"----------"
4290 DISPLAY AT(12,3):"DIENE PUNKTE SIND LEIDER "
4300 DISPLAY AT(14,3):"NEGATIV. WILLST DU"
4310 GOTO 1360
```

Invasion

für den Spectrum

Wenn Sie sich als Retter der Menschheit betätigen wollen, dann geben Sie dieses Listing ruhig in Ihren Spectrum ein. Werden Sie es wohl schaffen, die Invasion zu verhindern und das Raumschiff vorher abzuschießen? Dieses sehr schnelle Spiel fordert die ganze Grafik und den Tongenerator des Spectrums heraus. Ein gelungenes Spiel, das sich von selbst erklärt.

```
  5 BORDER 0: PAPER 0: INK 6
 10 RANDOMIZE : LET h=50
 20 CLS : LET s=0
 30 LET c=0: LET z=0
 40 LET x=16
 60 LET d=0: LET a=INT (RND*10)*2
 65 PRINT AT 21,0; INK 3;"5333744733388788747733384333377"
 70 IF d>=20 THEN  GO TO 2000
 80 LET a=a+RND: LET d=d+1
 85 IF a>=27 THEN  LET a=0
 90 IF d>=20 THEN  GO TO 2000
100 PRINT AT 20,29;"P ";AT 19,29;"o"
105 PRINT AT 20,0;"  o"
110 PRINT AT 20,x; INK 4;" mn "
130 PRINT AT d,a; INK 5;"stu"
140 PRINT AT d-1,0;"
```

EARTH DEFENCE

Versuchen Sie die Erde vor den fremden Raumschiffen zu retten.
Die Laserkanone wird mit den Tasten "S" und "D" nach links und nach rechts gesteuert. Geschossen wird mit der Leertaste.

```
1 RANDOMIZE
2 CALL CHAR(128,"FFFFFFFFFFFFFFFF")
3 CALL CHAR(125,"000000187E992400")
4 CALL CHAR(126,"0024242466666666")
5 CALL CHAR(127,"1818181818181818")
6 DIM P(32)
7 CALL CLEAR
8 S=0
9 CALL VCHAR(1,9,128,24)
10 CALL VCHAR(1,21,128,24)
11 Y1=1
12 Y2=1
13 Y3=1
14 X1=11
15 X2=15
16 X3=19
17 D1=1
18 D2=1
19 D3=1
20 X=15
21 CALL HCHAR(24,X,126)
22 CALL HCHAR(Y1,X1,125)
23 CALL HCHAR(Y2,X2,125)
24 CALL HCHAR(Y3,X3,125)
25 CALL CHAR(125,"000000187E992400")
26 CALL HCHAR(Y1,X1,32)
27 P(X1)=0
28 Y1=Y1+1
29 IF Y1=25 THEN 92
30 IF (X1=10)+(X1=20)THEN 31 ELSE 32
31 D1=-D1
32 X1=X1+D1
33 P(X1)=Y1
34 CALL HCHAR(Y1,X1,125)
35 CALL HCHAR(Y2,X2,32)
36 P(X2)=0
37 Y2=Y2+1
38 IF Y2=25 THEN 92
39 IF (X2=10)+(X2=20)THEN 40 ELSE 41
40 D2=-D2
41 X2=X2+D2
42 P(X2)=Y2
43 CALL HCHAR(Y2,X2,125)
44 CALL HCHAR(Y3,X3,32)
45 P(X3)=0
46 Y3=Y3+1
47 IF Y3=25 THEN 92
48 IF (X3=10)+(X3=20)THEN 49 ELSE 50
49 D3=-D3
50 X3=X3+D3
51 P(X3)=Y3
52 CALL HCHAR(Y3,X3,125)
53 CALL CHAR(125,"000000997E182400")
54 CALL KEY(0,KEY,STAT)
55 IF STAT=0 THEN 25
56 IF KEY=83 THEN 60
57 IF KEY=68 THEN 65
58 IF KEY=32 THEN 70
59 GOTO 25
60 IF X=11 THEN 25
61 CALL HCHAR(24,X,32)
62 X=X-1
63 CALL HCHAR(24,X,126)
64 GOTO 25
65 IF X=19 THEN 25
66 CALL HCHAR(24,X,32)
67 X=X+1
68 CALL HCHAR(24,X,126)
69 GOTO 25
70 CALL SOUND(250,-7,0)
71 CALL VCHAR(P(X)+1,X,127,23-P(X))
72 IF X1=X THEN 73 ELSE 78
73 CALL HCHAR(Y1,X1,32)
74 Y1=1
75 X1=INT(RND*9+11)
76 D1=INT(RND*3-1)
77 S=S+1
78 IF X2=X THEN 79 ELSE 84
79 CALL HCHAR(Y2,X2,32)
80 Y2=1
81 X2=INT(RND*9+11)
82 D2=INT(RND*3-1)
83 S=S+1
84 IF X3=X THEN 85 ELSE 90
85 CALL HCHAR(Y3,X3,32)
86 Y3=1
87 X3=INT(RND*9+11)
88 D3=INT(RND*3-1)
89 S=S+1
90 CALL VCHAR(P(X)+1,X,32,23-P(X))
91 GOTO 25
92 CALL CLEAR
93 PRINT "SCORE:";S::"HIT ANY KEY"
94 CALL KEY(0,KEY,STAT)
95 IF STAT=0 THEN 94 ELSE 7
```

Super-Miner

Zu Anfang des Spieles sind 4 verschiedene Tasten einzugeben, mit denen man sein Männchen durch die Mine steuern möchte.
Danach erklingt die Melodie des Liedes "Morning has broken". Drückt man nun eine Taste, so erscheint die erste Mine. In ihr sollen Diamanten gesammelt werden. Dazu stehen 99 Lufteinheiten zur Verfügung, die sich bei jedem Schritt um 2 verringern. An einigen Stellen in der Mine sind Sauerstofflaschen deponiert, an denen man seinen Luftvorrat um 40 Einheiten erhöhen kann.
Als Hindernisse sind Felsen und Türen in das Programm eingebaut. Will man an einem Felsen vorbei, so muß man diesen erst mit einer Spitzhacke zerschlagen, wozu 8 Lufteinheiten verbraucht werden. Durch die Türen gelangt man nur, wenn man mehr als 40 Einheiten besitzt. Sinkt die Zahl der Lufteinheiten unter 0, so zerfällt die Spielfigur zu Staub und wird mit einer Trauermelodie verabschiedet.
Gelingt es jedoch, ans anderen Ende der Mine mit heiler Haut zu gelangen, so kommt man in die nächste Runde. Haben Sie 500 Punkte erreicht, so erhalten Sie ein neues Männchen. Sind alle Männchen verbraucht und die erreichte Punktzahl ist höher als der angegebene Highscore, so können Sie Ihren Namen eintragen. Dieser wird in der nächsten Runde angezeigt. Ist die erreichte Punktzahl jedoch geringer, fragt der Computer nach einem neuen Spiel und ein neuer Versuch kann gestartet werden.
Das Spiel benötigt ein Extended-Basic-Modul und hat ca. 12.8 KB!

```
100 !************************
110 !************************
120 !*                      *
130 !*S U P E R - M I N E R*
140 !*----------------------*
150 !*    12.955 K-BYTE     *
160 !*                      *
170 !*   (C)1984 BY         *
180 !*   RALF MISCHOK       *
190 !*   GRUENER GRUND 35   *
200 !*   4403 SENDEN        *
210 !*                      *
220 !************************
230 !************************
240 ON BREAK NEXT :: ON WARNING NEXT :: RANDOMIZE :: CALL CLEAR :: CALL SCREEN(2
):: HI=299 :: N$="R MISCHOK"
250 DATA 3E2222222222223E,0202020202020202,3E02023E2020203E,3E02023E0202023E,222
2223E02020202
260 DATA 3E20203E0202023E,3E20203E2222223E,3E0202020202020202,3E22223E2222223E,3E2
2223E02020023E
270 RESTORE 250 :: FOR I=48 TO 57 :: READ A$ :: CALL CHAR(I,A$):: NEXT I
280 DATA 7C44447CC4C4C4C4,F8C8C8F8C4C4C4FC,F898808080808BF8,F8C4C4C4C4C4C4F8,F88
080F0C0C0C0F3
290 DATA FCC3C0F840404040,FCCCC0C0CCC4C4FC,888888FCCCCCCCCC,2020202060606060,787
8080808888C8F3
300 DATA 8890A0F8CCCCCCCC,4040404CC0C0C0FC,86CE86E6868686B6,84C4A494EC84B484,F86
8888889898BF8
310 DATA F8888888F8C3C2C0,F8888888989B9CFA,F8883888FCCCCCCC,F8888080F808C8F8,FC3
0303030303030
320 DATA 9898989898939EF8,8C8C8C8C8C4C2810,DADAC2C2C2D2EAC6,C8C8C87070C8C8C8,8C8
C4C2810101010
330 DATA F8880810204080F8
340 RESTORE 280 :: FOR I=65 TO 90 :: READ A$ :: CALL CHAR(I,A$):: NEXT I
350 CALL CHAR(33,"6060606000006060")
360 CALL CHAR(58,"0000606000006060")
370 CALL CHAR(64,"3C4299A1A199423C")
380 CALL CHAR(92,"00003E4908080808")
390 CALL CHAR(93,"08080808493E0000")
400 CALL CHAR(94,"0008040404FC0408")
410 CALL CHAR(95,"001020203F202010")
420 CALL CHAR(96,"202020FF040404FF")
430 CALL CHAR(104,"10307CFE7C301000")
440 CALL CHAR(112,"18183C5A18244242")
450 CALL CHAR(113,"181A7E5818542202")
460 CALL CHAR(114,"18587E1A182F4440")
470 CALL CHAR(115,"0018183C5A1E2442")
480 CALL CHAR(116,"000000283C7C7EFF")
490 CALL CHAR(117,"00000000103C7CFE")
500 CALL CHAR(118,"0000002424A55A3C")
510 CALL CHAR(119,"18187E7E18181818")
520 CALL CHAR(120,"1C7E7FFEFF7F7E34")
530 CALL CHAR(128,"1C044EAEFEAE4E0E")
540 CALL CHAR(136,"7E7E7E7E72767E7E")
550 FOR I=1 TO 8 :: CALL COLOR(I,15,1):: NEXT I
560 DISPLAY AT(2,4):"S U P E R - M I N E R"
570 DISPLAY AT(3,4):"----------------------"
580 DISPLAY AT(7,4):"(C)1984 BY RALF MISCHOK"
590 DISPLAY AT(9,4):"     GRUENER GRUND 35"
600 DISPLAY AT(11,4):"          4403 SENDEN"
610 DISPLAY AT(15,4):"KEY LEFT:" :: DISPLAY AT(17,4):"KEY RIGHT:" :: DISPLAY AT(
19,4):"KEY UP:" :: DISPLAY AT(21,4):"KEY DOWN:"
620 ACCEPT AT(15,15)BEEP SIZE(1):L$ :: ACCEPT AT(17,15)BEEP SIZE(1):R$
630 ACCEPT AT(19,15)BEEP SIZE(1):O$ :: ACCEPT AT(21,15)BEEP SIZE(1):U$
640 IF L$="" OR R$="" OR O$="" OR U$="" THEN 620
650 DATA 300,262,300,330,300,392,600,523,600,587,300,494,300,440,300,392,600,440
,150,494,150,440
660 DATA 300,392,300,262,300,294,300,330,600,392,600,440,300,392,300,294,600,294
```

AIR	= Anzahl der Lufteinheiten
BO	= Anzahl der Bonusmännchen
D	= Anzahl der Diamanten
F	= Anzahl der Felsen
HI	= Highscore
I.J.K.L	= Laufvariablen
MEN	= Anzahl der Männchen
P	= Punktezahl
PD	= Punkte für Diamanten
PS	= Punkte für Sauerstofflaschen
PT	= Punkte für Türen
R	= Anzahl der Runden
SA	= Anzahl der Sauerstofflaschen
T	= Anzahl der Türen
X Y	= Position des Männchens
Z	= Variable für GCHAR-Statement
A$	= Selbstdef. Buchstaben und Zahlen
L$	= Taste links
O$	= Taste oben
R$	= Taste rechts
U$	= Taste unten
N$	= Name für Highscore

Farb-Verteilung

Buchstaben	= Grau
Diamant	= Hellgelb
Felsen	= Hellgrün
Hintergrund	= Schwarz
Männchen	= Cyan
Männchen (tot)	= Cyan
Mauer	= Mittelrot/Weiß
Sauerstofflasche	= Dunkelgelb
Spitzhacke	= Grau
Tür	= Mittelgrün
Zahlen	= Grau

Punkteverteilung
1. Runde:

Diamant 3 Punkte
Sauerstofflasche 1Punkt
Tür 2 Punkte
Jede weitere Runde bekommt 1 Punkt dazu.

```
,600,294,300,392
670 DATA 300,330,300,392,600,523,600,440,300,392,300,330,300,294,600,262,600,294
,300,330,300,294
680 DATA 300,330,600,392,600,440,300,294,300,330,300,294,600,262,100,43000
690 RESTORE 650 :: FOR I=1 TO 40 :: READ J,K :: CALL SOUND(J,K,0,K-2,10,K+2,10):
: NEXT I
700 DISPLAY AT(24,1):"PRESS ANY KEY TO START GAME!" :: FOR I=1 TO 100 :: CALL KE
Y(0,K,S):: IF S=0 THEN 710 ELSE 730
710 NEXT I
720 DISPLAY AT(24,1):"" :: GOTO 690
730 CALL CLEAR :: R=0 :: SA=6 :: P=0  :: BO=0 :: MEN=3 :: PD=3 :: PT=2 :: PS=1
740 R=R+1
750 CALL CLEAR :: FOR I=9 TO 14 :: CALL COLOR(I,2,1):: NEXT I :: CALL COLOR(8,15
,1):: X=5  :: Y=5 :: AIR=99
760 CALL HCHAR(5,1,96,640)
770 DISPLAY AT(1,1):"SCORE:";P :: DISPLAY AT(1,13):"AIR:";AIR :: DISPLAY AT(1,22
):"MEN:";MEN
780 DISPLAY AT(3,1):"HIGHSCORE:";HI;"BY ";N$
790 ON R GOSUB 1580,1650,1730,1810,1890,1970,1580,1650,1730,1810,1890,1970,1580,
1650,1730,1810,1890,1970,1580,1650,1730,1810,1890,1970
800 CALL COLOR(8,15,1,9,16,7,10,12,1,11,3,1,12,4,1,13,11,1,14,3,1)
810 FOR I=1 TO SA
820 J=INT(RND*17)+6 :: K=INT(RND*25)+2 :: CALL GCHAR(J,K,Z):: IF Z<>32 THEN 820
:: CALL HCHAR(J,K,128)
830 NEXT I
840 FOR I=1 TO T
850 J=INT(RND*17)+6 :: K=INT(RND*25)+2 :: CALL GCHAR(J,K,Z):: IF Z<>32 THEN 850
:: CALL HCHAR(J,K,136)
860 NEXT I
870 FOR I=1 TO F
880 J=INT(RND*17)+6 :: K=INT(RND*25)+2 :: CALL GCHAR(J,K,Z):: IF Z<>32 THEN 880
:: CALL HCHAR(J,K,120)
890 NEXT I
900 FOR I=1 TO D
910 J=INT(RND*17)+6 :: K=INT(RND*25)+2 :: CALL GCHAR(J,K,Z):: IF Z<>32 THEN 910
:: CALL HCHAR(J,K,104)
920 NEXT I
930 CALL HCHAR(X,Y,112):: CALL HCHAR(4,5,30):  CALL HCHAR(4,27,31)
940 FOR I=1 TO 15  :: CALL SOUND(-200,600,0,595,9,605,9):: CALL SOUND(-200,700,0,
695,9,705,9):: NEXT I
950 CALL KEY(0,K,S):: IF S=0 THEN 960 ELSE 970
960 CALL HCHAR(X,Y,112):: AIR=AIR-1 :: DISPLAY AT(1,17)SIZE(4):AIR :: FOR I=1 TO
 70 :: NEXT I :: IF AIR<1 THEN 1070 ELSE 950
970 DISPLAY AT(1,7)SIZE(5):P :: DISPLAY AT(1,17)SIZE(4):AIR :: IF AIR<1 THEN 107
0 :: DISPLAY AT(1,26)SIZE(2):MEN
980 IF K=ASC(O$)THEN 990 ELSE IF K=ASC(U$)THEN 1010 ELSE IF K=ASC(R$)THEN 1030 E
LSE IF K=ASC(L$)THEN 1050 ELSE 950
990 CALL GCHAR(X-1,Y,Z):: IF Z=30 OR Z=96 THEN 950 ELSE IF Z=31 THEN 1290 ELSE I
F Z=104 THEN P=P+PD ELSE IF Z=120 THEN 1380 ELSE IF Z=128 THEN 1170 ELSE IF Z=13
6 AND AIR<40 THEN 950 ELSE IF Z=136 THEN P=P+PT
1000 AIR=AIR-2 :: CALL HCHAR(X,Y,32):: X=X-1  :: CALL SOUND(-50,350,0,348,9,352,9
):: CALL HCHAR(X,Y,112):: GOTO 950
1010 CALL GCHAR(X+1,Y,Z):: IF Z=96 THEN 950 ELSE IF Z=104 THEN P=P+PD ELSE IF Z=
120 THEN 1430 ELSE IF Z=128 THEN 1200 ELSE IF Z=136 AND AIR<40 THEN 950 ELSE IF
Z=136 THEN P=P+PT
1020 AIR=AIR-2 :: CALL HCHAR(X,Y,32):: X=X+1  :: CALL SOUND(-50,450,0,448,9,452,9
):: CALL HCHAR(X,Y,112):: GOTO 950
1030 CALL GCHAR(X,Y+1,Z):: IF Z=96 THEN 950 ELSE IF Z=104 THEN P=P+PD ELSE IF Z=
120 THEN 1480 ELSE IF Z=128 THEN 1230 ELSE IF Z=136 AND AIR<40 THEN 950 ELSE IF
Z=136 THEN P=P+PT
1040 AIR=AIR-2 :: CALL HCHAR(X,Y,32):: Y=Y+1 :: CALL SOUND(-50,400,0,398,9,402,9
):: CALL HCHAR(X,Y,113):: GOTO 950
1050 CALL GCHAR(X,Y-1,Z):: IF Z=96 THEN 950 ELSE IF Z=104 THEN P=P+PD ELSE IF Z=
120 THEN 1530 ELSE IF Z=128 THEN 1260 ELSE IF Z=136 AND AIR<40 THEN 950 ELSE IF
Z=136 THEN P=P+PT
1060 AIR=AIR-2 :: CALL HCHAR(X,Y,32):: Y=Y-1  :: CALL SOUND(-50,500,0,498,9,502,9
):: CALL HCHAR(X,Y,114):: GOTO 950
1070 J=350 :: FOR I=115 TO 119 :: CALL HCHAR(X,Y,I):: CALL SOUND(-600,J,0):: FOR
 K=1 TO 110 :: NEXT K :: J=J-50 :: NEXT I
1080 DATA 500,196,500,196,100,196,500,196,500,233,100,220,250,220,250,196,250,19
6,250,185,1000,196,100,43000
1090 RESTORE 1080 :: FOR I=1 TO 11 :: READ J,K :: CALL SOUND(J,K,0,K-1,9,K+1,9):
: NEXT I
1100 FOR I=1 TO 800 :: NEXT I :: CALL HCHAR(X,Y,32):: MEN=MEN-1 :: IF MEN<1 THEN
 1120 ELSE X=5 :: Y=5 :  AIR=99
1110 DISPLAY AT(1,7)SIZE(5):P :: DISPLAY AT(1,17)SIZE(4):AIR :: DISPLAY AT(1,26)
SIZE(2):MEN :: GOTO 930
1120 CALL COLOR(8,15,1) : CALL CLEAR :: DISPLAY AT(10,4):"G A M E  O V E R !  !
" :: DISPLAY AT(14,5):"YOUR SCORE:";P :: IF P>HI THEN 1130 :: DISPLAY AT(16,1):"
HIGHSCORE:";HI;"BY ";N$ :: GOTO 1140
1130 DISPLAY AT(18,5):"HIGHSCORE!" :: HI=P :: DISPLAY AT(20,5):"YOUR NAME:" :  A
CCEPT AT(20,16)SIZE(5):N$
1140 DISPLAY AT(23,5):"YOU WANT A NEW GAME (Y/N)?"
1150 CALL KEY(0,K,S):: IF S=0 THEN 1150
1160 IF K=89 THEN CALL CLEAR :: GOTO 550 ELSE IF K=78 THEN CALL CLEAR :: END ELS
E 1150
1170 CALL HCHAR(X,Y,32):: X=X-1 :: CALL HCHAR(X,Y,112)
1180 FOR I=300 TO 700 STEP 10 :: CALL SOUND(-100,I,0):: DISPLAY AT(1,17)SIZE(4):
AIR :: AIR=AIR+1 :: NEXT I
1190 P=P+PS :: GOTO 950
1200 CALL HCHAR(X,Y,32):: X=X+1 :: CALL HCHAR(X,Y,112)
1210 FOR I=300 TO 700 STEP 10 :: CALL SOUND(-100,I,0):: DISPLAY AT(1,17)SIZE(4):
AIR :: AIR=AIR+1 :: NEXT I
1220 P=P+PS :: GOTO 950
1230 CALL HCHAR(X,Y,32):: Y=Y+1 :: CALL HCHAR(X,Y,112)
```

```
1240 FOR I=300 TO 700 STEP 10 :: CALL SOUND(-100,I,0):: DISPLAY AT(1,17)SIZE(4):
AIR :: AIR=AIR+1 :: NEXT I
1250 P=P+PS :: GOTO 950
1260 CALL HCHAR(X,Y,32):: Y=Y-1 :: CALL HCHAR(X,Y,112)
1270 FOR I=300 TO 700 STEP 10 :: CALL SOUND(-100,I,0):: DISPLAY AT(1,17)SIZE(4):
AIR :: AIR=AIR+1 :: NEXT I
1280 P=P+PS :: GOTO 950
1290 FOR I=1 TO 2 :: FOR J=220 TO 550 STEP 10 :: CALL SOUND(-200,J,0,J-2,9,J+2,9
):: NEXT J :: FOR J=550 TO 220 STEP -10 :: CALL SOUND(-200,J,0,J-2,9,J+2,9):: NE
XT J :: NEXT I
1300 CALL CLEAR :: DISPLAY AT(10,5):"ROUND";R;"OVER!" :: DISPLAY AT(12,5):"YOUR
SCORE:";P :: DISPLAY AT(14,5):"BONUS:";INT(AIR/5):: P=P+INT(AIR/5)
1310 DISPLAY AT(16,5):"YOUR SCORE:";P :: IF P>=500 AND BO=0 OR P>=1000 AND BO=1
OR P>=2000 AND BO=2 OR P>=2500 AND BO=3 THEN 1320 ELSE 1350
1320 BO=BO+1 :: MEN=MEN+1
1330 DISPLAY AT(18,5):"BONUS-MAN!" :: CALL SOUND(-200,600,0,500,0,700,0):: FOR I
=1 TO 100 :: NEXT I :: DISPLAY AT(18,5):"" :: FOR J=1 TO 80 :: NEXT J
1340 CALL KEY(0,K,S):: IF S=0 THEN 1330
1350 CALL KEY(0,K,S):: IF S=0 THEN 1350
1360 PO=PO+1 :: PT=PT+1 :: PS=PS+1 :: SA=SA-1 :: IF SA<1 THEN SA=1
1370 CALL CLEAR :: FOR I=6 TO 14 :: CALL COLOR(I,2,1):: NEXT I :: GOTO 740
1380 CALL HCHAR(X,Y,32):: X=X-1 :: CALL HCHAR(X,Y,112)
1390 FOR I=1 TO 8
1400 CALL HCHAR(X,Y,94):: CALL SOUND(100,-7,0):: CALL HCHAR(X,Y,92):: CALL SOUND
(100,-6,0):: AIR=AIR-1 :: DISPLAY AT(1,17)SIZE(4):AIR
1410 IF AIR<1 THEN 1070
1420 NEXT I :: CALL HCHAR(X,Y,112):: GOTO 950
1430 CALL HCHAR(X,Y,32):: X=X+1 :: CALL HCHAR(X,Y,112)
1440 FOR I=1 TO 8
1450 CALL HCHAR(X,Y,94):: CALL SOUND(100,-7,0):: CALL HCHAR(X,Y,93):: CALL SOUND
(100,-6,0):: AIR=AIR-1 :: DISPLAY AT(1,17)SIZE(4):AIR
1460 IF AIR<1 THEN 1070
1470 NEXT I :: CALL HCHAR(X,Y,112):: GOTO 950
1480 CALL HCHAR(X,Y,32):: Y=Y+1 :: CALL HCHAR(X,Y,112)
1490 FOR I=1 TO 8
1500 CALL HCHAR(X,Y,92):: CALL SOUND(100,-7,0):: CALL HCHAR(X,Y,94):: CALL SOUND
(100,-6,0):: AIR=AIR-1 :: DISPLAY AT(1,17)SIZE(4):AIR
1510 IF AIR<1 THEN 1070
1520 NEXT I :: CALL HCHAR(X,Y,112):: GOTO 950
1530 CALL HCHAR(X,Y,32):: Y=Y-1 :: CALL HCHAR(X,Y,112)
1540 FOR I=1 TO 8
1550 CALL HCHAR(X,Y,92):: CALL SOUND(100,-7,0):: CALL HCHAR(X,Y,95):: CALL SOUND
(100,-6,0):: AIR=AIR-1 :: DISPLAY AT(1,17)SIZE(4):AIR
1560 IF AIR<1 THEN 1070
1570 NEXT I :: CALL HCHAR(X,Y,112):: GOTO 950
1580 DATA 7,2,16,5,5,3,7,8,16,7,14,16,7,19,16,7,25,16,5,27,3,7,31,16
1590 RESTORE 1580 :: FOR I=1 TO 8 :: READ J,K,L :: CALL VCHAR(J,K,32,L):: NEXT I
1600 DATA 7,2,30,11,14,6,12,11,4,12,19,4,13,2,7,13,25,7,16,2,7,16,25,7,17,11,4,1
7,19,4

1610 DATA 18,14,6,22,2,30
1620 RESTORE 1600 :: FOR I=1 TO 12 :: READ J,K,L :: CALL HCHAR(J,K,32,L):: NEXT
I
1630 T=5 :: D=15 :: F=12
1640 RETURN
1650 DATA 5,5,12,11,7,9,19,12,5,14,13,6,10,14,5,14,16,4,6,17,6,19,19,4,6,23,11,4
,27,3,14,27,6
1660 DATA 19,24,4
1670 RESTORE 1650 :: FOR I=1 TO 12 :: READ J,K,L :: CALL VCHAR(J,K,32,L):: NEXT
I
1680 DATA 7,14,16,8,2,10,10,12,3,11,2,8,11,21,11,14,10,20,16,3,2,17,7,7,17,16,4,
19,2,12
1690 DATA 19,16,4,19,24,7,22,15,10,23,12,4
1700 RESTORE 1680 :: FOR I=1 TO 14 :: READ J,K,L :: CALL HCHAR(J,K,32,L):: NEXT
I
1710 T=5 :: D=20 :: F=18
1720 RETURN
1730 DATA 13,2,5,17,4,6,5,5,9,13,9,5,19,9,4,12,11,2,12,14,3,17,14,6,7,17,6,14,17
,5,7,20,2
1740 DATA 12,20,12,18,22,5,8,23,5,14,24,3,14,26,5,5,27,4,8,28,4,17,29,7,7,31,10
1750 RESTORE 1730 :: FOR I=1 TO 19 :: READ J,K,L :: CALL VCHAR(J,K,32,L):: NEXT
I
1760 DATA 7,3,16,8,21,1,11,3,3,12,14,10,13,5,7,14,24,8,17,2,13,10,14,10,18,16,5
,19,6,7
1770 DATA 22,3,27,14,13,6,15,20,5,18,26,6
1780 RESTORE 1760 :: FOR I=1 TO 14 :: READ J,K,L :: CALL HCHAR(J,K,32,L):: NEXT
I
1790 T=3 :: D=22 :: F=20
1800 RETURN
1810 DATA 16,3,6,5,5,12,12,7,10,8,11,11,16,14,8,6,17,13,14,20,6,21,22,3,11,23,7,
14,26,10
1820 DATA 5,27,8,9,31,15
1830 RESTORE 1810 :: FOR I=1 TO 12 :: READ J,K,L :: CALL VCHAR(J,K,32,L):: NEXT
I
1840 DATA 6,2,18,7,24,8,9,16,7,10,3,11,10,26,6,12,16,13,13,2,8,14,14,5,16,23,9,1
7,2,6
1850 DATA 17,9,13,19,29,3,20,6,11,21,25,4,23,11,21
1860 RESTORE 1840 :: FOR I=1 TO 15 :: READ J,K,L :: CALL HCHAR(J,K,32,L):: NEXT
I
1870 T=3 :: D=20 :: F=18
1880 RETURN
1890 DATA 9,3,7,5,5,3,20,4,4,6,7,13,18,9,6,9,10,4,6,13,8,17,13,4,6,16,18,9,20,12
,14,22,10
```

```
1900 DATA 6,24,11,5,27,5,10,27,14
1910 RESTORE 1890 :: FOR I=1 TO 14 :: READ J,K,L :: CALL VCHAR(J,K,32,L)::
I
1920 DATA 6,3,8,7,12,8,7,22,9,10,2,13,12,15,16,15,5,8,15,14,4,15,19,12,18,2,9,18
,12,20
1930 DATA 21,3,5,22,9,15,22,26,6
1940 RESTORE 1920 :: FOR I=1 TO 13 :: READ J,K,L :: CALL HCHAR(J,K,32,L)::
I
1950 T=9 :: D=25 :: F=22
1960 RETURN
1970 DATA 8,2,9,16,4,5,22,4,2,5,5,8,18,7,3,7,8,5,11,9,13,16,12,8,6,13,11,18,14,4
,11,16,8
1980 DATA 21,16,3,20,18,4,6,19,15,9,21,2,12,21,12,9,23,4,8,24,2,12,24,8,9,25,2,1
2,26,6
1990 DATA 5,27,8,20,28,4,7,30,3,14,31,7,22,31,2
2000 RESTORE 1970 :: FOR I=1 TO 26 :: READ J,K,L :: CALL VCHAR(J,K,32,L)::
I
2010 DATA 6,9,5,6,19,9,7,5,5,7,13,5,7,27,4,8,2,4,9,10,4,9,15,11,11,8,3,11,13,7,1
2,2,5
2020 DATA 12,12,2,12,19,3,12,23,8,14,5,3,14,29,3,15,15,2,15,21,4,16,2,12,16,16,4
,16,24,6
2030 DATA 18,12,5,10,21,2,13,28,4,19,3,2,19,9,4,19,24,2,20,4,4,20,28,4,21,21,8,2
2,4,6
2040 DATA 23,12,5,23,18,6,23,28,4
2050 RESTORE 2010 :: FOR I=1 TO 34 :: READ J,K,L :: CALL HCHAR(J,K,32,L)::
I
2060 T=10 :: D=25 :: F=22
2070 RETURN
```

Impressum

CPU (Computer programmiert zur Unterhaltung) erscheint monatlich im:
Roeske Verlag Eschwege

Herausgeber:
Ralph Roeske

Redaktion:
Ralph Roeske (Chefredakteur verantwortlich)
Christian Wunder, Gertrud Marx-Fischer

Freie Mitarbeiter:
R. v. d. Steen, V. Bock, H. Mischck, M. Kurtz, B. Weber, N. Adam, E. Wedding, M. Bormann, M. Wellenhofer, A. Woyca, O. Hilgendorf.

Herstellung:
Roeske Verlag Eschwege

Satz und Reproduktion:
Roeske Verlag Eschwege

Druck:
Vogt GmbH, 3436 Hessisch Lichtenau

Vertrieb:
Inland (Groß-, Einzel- und Bahnhofsbuchhandel), sowie Österreich und Schweiz:
Verlagsunion
Friedrich-Bergius-Straße 7
6200 Wiesbaden
Tel.: 06121/2660

Anfragen nicht an Vertrieb oder Druckerei, sondern nur an den Verlag!

Anschrift:
Roeske Verlag
CPU
Fuldaer Straße 6
3440 Eschwege
Tel. Sa. Nr. 05651/9558

Anzeigenleitung:
W & M
Agentur für Werbung
und Marketing GmbH
Postfach 111341
4000 Düsseldorf 1
Tel.: 0211/555 56
Tx.: 8584109

Erscheinungsweise:
Erstverkaufstag von CPU ist Mitte des Monats.

Bezugspreise:
Einzelheft: 5,50 DM
Abonnement: Inland 55,- DM im Jahr (12 Ausgaben)
Ausland: Europa 80,- DM USA 110,- DM

Anzeigenpreise:
Es gilt die Anzeigenpreisliste Nr. 1 vom 1. 8. 1983. Bitte Media-Unterlagen anfordern.

Urheberrecht:
Alle in CPU veröffentlichen Beiträge sind urheberrechtlich geschützt. Alle Rechte, auch Übersetzungen, vorbehalten.
Reproduktionen jeder Art (Fotokopie, Microfilm, Erfassung in Datenverarbeitungsanlagen usw.) bedürfen der schriftlichen Genehmigung des Verlags. Alle veröffentlichte Software wurde von Mitarbeitern des Verlages oder von freien Mitarbeitern erstellt.
Aus ihrer Veröffentlichung kann nicht geschlossen werden, daß die beschriebenen Lösungen oder Bezeichnungen frei von Schutzrechten sind.

Autoren, Manuskripte:
Der Verlag nimmt Manuskripte und Software zur Veröffentlichung gerne entgegen.
Sollte keine andere Vereinbarung getroffen sein, so gehen wir davon aus, daß Sie mit einem Honorar von DM 100,- pro gedr. Seite im Heft, einverstanden sind.
Bei Zusendung von Manuskripten und Software, erteilt der Autor dem Verlag die Genehmigung zum Abdruck und Versand der veröffentlichten Programme auf Datenträger.
Rücksendung erfolgt nur gegen Erstattung der Unkosten. Zusendungen von Software zur Veröffentlichung soll bitte folgendes enthalten:
Kopierfähige Kassette oder Diskette mit dem Programm (Computer-Bezeichnung), von Drucker erstelltes Listing oder Serie von Bildschirmfotos (keine Schreibmaschinenlistings), evtl. Bildschirmfotos von einem Probelauf, ausführliche Programmbeschreibung. (Erklärung der Variablenliste, Beschreibung des Bildschirmaufbaues, Farbe, Grafik usw.)
Für eingesandte Programmunterlagen kann keinerlei Haftung übernommen werden.
IVW geprüft.

mcc Laden
Micro Computer Christ
Rathausstraße 4, 2300 Kiel 1
Telefon (04 31) 9 63 76
APPLE
ATARI
BASIS
Commodore
DAI
EACA
OSBORNE
SHARP
TANDY

Der COMPUTER Laden
TANDY APPLE ATARI OLYMPIA BOSS
EPSON OLYMPIA CENTRONICS PLOTTER
LAUFWERKE MONITORE DISKETTEN
INDIVICUALSOFTWARE INTERFACES
Wuppertaler Str. 98
5650 Solingen 1
Telefon (0 21 22) 1 66 37

BUCHHANDLUNG
GONSKI
Fachbücher + Fachzeitschriften für Mikrocomputer
Gertrudenstraße 2–4, (Ecke Neumarkt)
5000 Köln 1, Telefon (02 21) 21 05 28

TA Micro Computer-Laden
Am Marstall 20/22
3000 Hannover 1
Telefon (0511) 14629
alphatronic
trend DATA

ZX SPECTRUM

Speichererweiterung auf 48 K	89,– DM
Speichererweiterung auf 80 K	189,– DM
programmierbares Joystick-Interface (für alle Spiele)	110,– DM
Joystick-Interface (kempston-Compatibel)	53,– DM
Quickshot-Joystick	39,– DM
Light-Pen	84,– DM
Profi-Tastatur mit Zwölfertastenfeld	188,– DM

Alle Preise incl. MwSt. + Porto + NN
Ausführliches Info gegen 2,50 DM Rückporto.

COMPUTER & MEDIENTECHNIK · HEINZ MEYER
FAHSERSTR. 58 · 4060 VIERSEN 1 · TEL. 0 21 62/2 29 64

Astro-Fighter

für den TI-99/4A

Grundversion

Spielanleitung:
Astro-Fighter ein Superspiel in TI-Basic mit toller Grafik und Spitzen-Sound.
In der ersten Spielphase greifen feindliche Ufos an und versuchen, Ihr Raumschiff zu zerstören. Dabei ist schnelle Reaktion erforderlich!
Zur Abwehr der Ufos steht ein Laser (Leertaste) und ein Schutzschirm (Taste B) zur Verfügung. Ein Schuß mit dem Laser kostet 250 und ein Schutzschirm 500 Energieeinheiten, daher ist mit dem Laser und Schutzschirm sparsam umzugehen, da nur eine Gesamtenergie vor 9000 zur Verfügung steht. Zu Beginn des Spieles besitzen Sie 3 Raumschiffe, die mit "Q" nach oben und mit "Z" nach unten gesteuert werden.
In der zweiten Spielphase muß das Raumschiff auf einem Mutterschiff landen und dabei einen Drift ausgleichen, welcher besondere Geschicklichkeit erfordert. Falls Ihnen die Restenergie der ersten Phase zur Landung reicht, werden Sie mit einem Bonus von 300 Punkten belohnt. Bei 3000 bzw. 6000 Punkten erhalten Sie ein neues Raumschiff. Danach beginnt das Spiel von vorne.
Haben Sie alle drei Raumschiffe verloren oder Ihre ganze Energie verbraucht, ist das Spiel beendet. Sie können dann Ihren Namen in die Highscoretabelle eintragen. Diese enthält die Namen und Punktzahlen der 10 besten Spieler und kann nach Spielende durch "H" aufgerufen werden. Ansonsten drückt man hier "S" für Neustart oder "E" zum Beenden.

Variablenliste:
M,N,O,P,Q,R,S Zählvariablen der Schleifen
XS,Z Notenlänge beim Sound
A(x),H(x),D(x),FIS(x),CIS(x) Notenwerte
G(x),C(x),F,B,C,K,S Variablen zur Tastaturabfrage
DF Anzahl der Raumschiffe
ENERGIE Energievorrat
SCORE Punktzahl
FEI Anzahl der erschienenen Ufos
X,Y,Q Koordinaten der Raumschiffe bzw. Ufos
B,BI Hilfsvariablen zur Bewegung
R Code des vorhandenen Ufos (o wenn kein Ufo)
ZEI Angabe ob Laser ein- oder zweistrahlig ist
YS Textausgabezeile im Bildschirm
XD,YD Schub des Raumschiffes in der 2. Phase
SC1 Gibt an, welcher Bonus schon gegeben wurde
HS(x) Highscoretabelle (Punkte)
H$(x) Highscoretabelle (Namen)
F$ Zeichenkette zur Auswahl der Ufos
E$,S$,D$ Zeichenketten zur Generierung der Punktezeile
N$ Eingabe des Namens

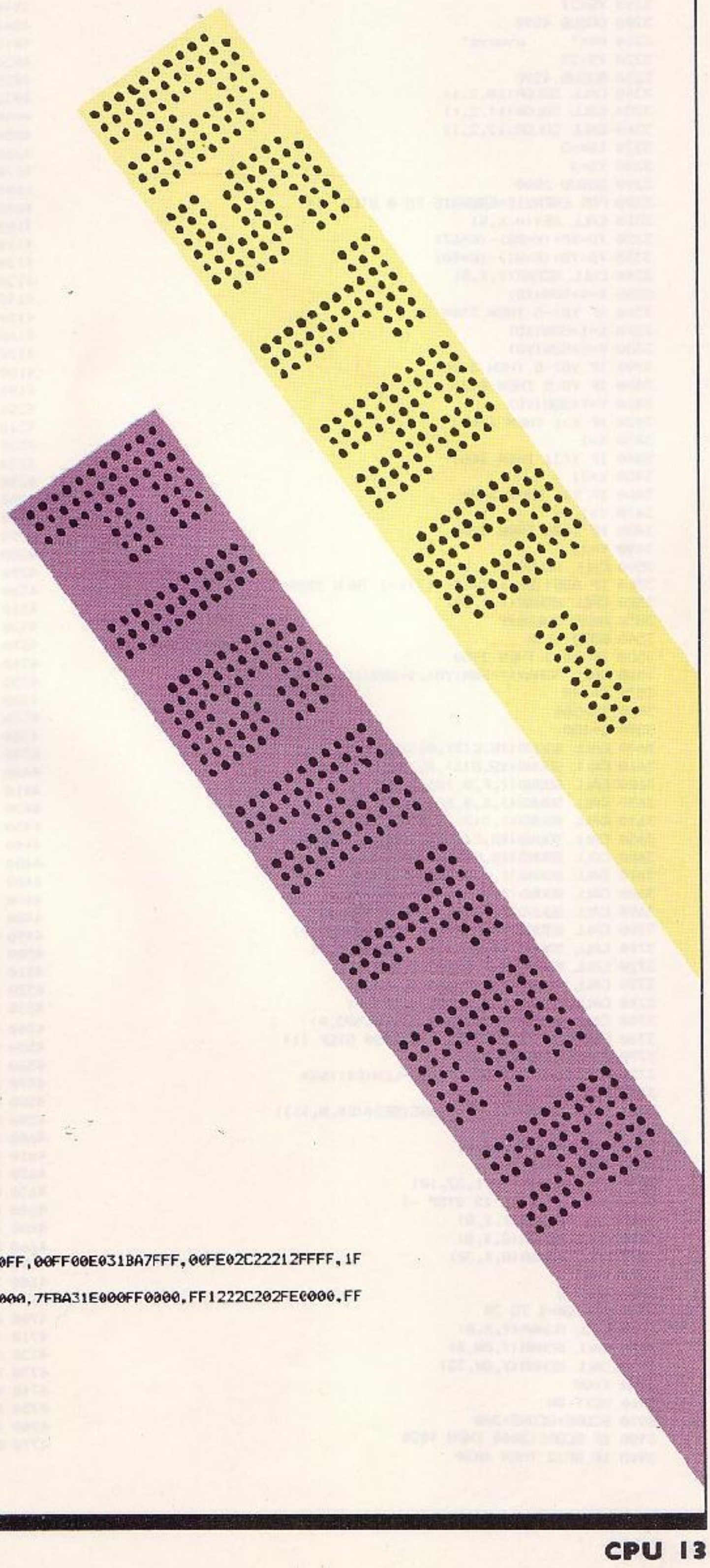
Programmteile:

Zeile	Aufgabe
140-460	Zeichendefinierung für 1. Phase
470-540	Zeichendefinierung für Vorspann
550-680	Hauptroutine für Vorspann
690-950	Definierung der Notenwerte
960-1590	E-Rock
1600-1680	Triebwerkssound im Vorspann
1690-1750	Warten auf Startbefehl
1760-1840	Variablendefinierung für 1. Phase
1850-1930	Festlegung der Vorder- und Hintergrundfarben
1940-1990	Erzeugung der Ansicht
2000-2090	Steuerung Raumschiff
2100-2200	Ermittlung des nächsten Ufos und dessen Position
2210-2300	Steuerung Ufo
2310-2480	Schuß des Ufos und Schutzschirmaufbau
2490 2600	Schuß des Raumschiffes
2610-2790	Game Over
2800-2940	Generierung der Punktezeile
2950-3030	Crash des Raumschiffes
3040-3290	Vorbereitung 2. Phase
3300-3540	Bewegung 2. Phase
3550-3750	Siegesfanfare
3760-3830	Erhöhen der Energie
3840-3960	Raumschiff startet
3970-4080	Bonuswertung
4090-4420	Aufstellen der Highscoretabelle
4430-4620	Eintrag in Highscoretabelle
4630-4770	Crash bei leerem Tank in der 2. Phase

```
100 REM    ASTRO - FIGHTER          (C) ARBEITSKREIS'                SOFTWARE
1984
110 REM                             PROGRAM: Gg.W.Seefried      SOUND  : J. Flie
r)          GRAFIK : A. Reiss
120 CALL CLEAR
130 CALL SCREEN(5)
140 DATA 7E464A4A5252627E.0C142464040404 0E,3C44B0B0B03062 7E./E42021E1A02427E.0C1
828497F0R0B1C.7E42407E0A064278
150 DATA 7E42407C4646467E.7E460E1020202070.3C24247E4343437E.7E62623E0202427E
160 DATA 00007E027E427E00.40407E4242427E00.00007E4040407E00.02027E4242427E00.000
07E427E407E00.1E10107E10107000
170 DATA 00007E427E027E00.40407E4242424200.1000101010101000.0800030808102000.485
060504E444200.1010101010100800
180 DATA 0000 7E4A4A4A4A00.0000 7E4242424200.0000 7E4242427E00.0000 7E427E4040000.000
07E427E020200.00007E4040404000
190 DATA 00007E407E027E00.10107E1010101000.0000424242427E00.0000424224241800.000
0425252527E00.0000422418244200
200 DATA 000042427E101000.00007E04 8207E00
210 CALL CHAR(33,"3FF80B3F3F0BFB3F")
220 CALL CHAR(34,"2842841248118412")
230 CALL CHAR(35,"0008081C7F1C080B")
240 CALL CHAR(42,"FF1B3C7E7E3C1BFF")
250 CALL CHAR(43,"0000183CFF240000")
260 CALL CHAR(60,"183C5AFF24428100")
270 CALL CHAR(61,"0000407E7F000000")
280 CALL CHAR(62,"7C30FF78FF307C00")
290 CALL CHAR(91,"0000000010000000")
300 CALL CHAR(38,"4242DB99FF99BDA5")
310 CALL CHAR(92,"00000000FF000000")
320 CALL CHAR(93,"0000FF00FF000000")
330 CALL CHAR(94,"24007E027E427E00")
340 CALL CHAR(95,"00FF0000000FF00")
350 CALL CHAR(64,"3C4299A1A199423C")
360 CALL CHAR(123,"2400424242427E00")
370 CALL CHAR(124,"42007E4242427E00")
380 CALL CHAR(126,"0001010101000100")
390 FOR N=48 TO 57
400 READ A$
410 CALL CHAR(N,A$)
420 NEXT N
430 FOR N=65 TO 90
440 READ A$
450 CALL CHAR(N,A$)
460 NEXT N
470 RANDOMIZE
480 DATA 0003E3F02020202,00C000FF0009F989,000000FF00001312,000000FF0000E404,000
000FF00000907,000000F000001070
490 DATA 020202023F221E00,B9F988B88F00FF00,1213A243FF00FF00,043404E7FF00FF00,090
909C9FF00FF00,503010101010F000
500 RESTORE 480
510 FOR N=97 TO 168
520 READ A$
530 CALL CHAR(N,A$)
540 NEXT N
550 PRINT "     (MUSIC BY abcdef)                    ghijkl":::
560 PRINT "   GENERAL COMPUTER-CLUB"::
570 PRINT "  A R B E I T S K R E I S"
580 PRINT :::"      S O F T W A R E"
590 PRINT ::::"        PR^SENTIERT'::::
600 GOSUB 690
610 CALL CLEAR
620 CALL SCREEN(7)
630 PRINT "*******************************                **      A  S
T  R  O       *"
640 PRINT "*                          **   F  I  G  H  T  E  R    **
          *"
650 PRINT "*                          ** @ ARBEITSKREIS SOFTWARE  **    NEUHOF A
N DER ZENN    *"
660 PRINT "*                          **          1984           **
          *"
670 PRINT "*******************************":::::
680 GOTO 1690
690 GOSUB 1600
700 Z=150
710 A(1)=110
720 A(2)=220
730 A(3)=440
740 A(4)=880
750 H(1)=123
760 H(2)=247
770 H(3)=494
```

```
780 H(4)=988
790 D(1)=147
800 D(2)=294
810 D(3)=587
820 FIS(1)=185
830 FIS(2)=370
840 FIS(3)=740
850 F=698
860 E=659
870 CIS(1)=139
880 CIS(2)=277
890 CIS(3)=554
900 G(1)=196
910 G(2)=392
920 G(3)=784
930 B=932
940 C(3)=523
950 C(4)=1047
960 FOR O=1 TO 2
970 FOR N=1 TO 2
980 CALL SOUND(Z,H(1),0,H(2),0,H(3),0)
990 CALL SOUND(Z,H(1),0,H(2),0,H(3),0)
1000 CALL SOUND(Z,FIS(3),0,743,0,737,0)
1010 CALL SOUND(Z,H(3),0,496,0,492,0)
1020 CALL SOUND(Z,E,0,661,0,657,0)
1030 CALL SOUND(Z,H(3),0,496,0,492,0)
1040 CALL SOUND(Z,CIS(3),0,556,0,552,0)
1050 CALL SOUND(Z,D(3),0,589,0,585,0)
1060 NEXT N
1070 FOR M=1 TO 2
1080 CALL SOUND(Z,A(1),0,A(2),0,A(3),0)
1090 CALL SOUND(Z,A(1),0,A(2),0,A(3),0)
1100 CALL SOUND(Z,FIS(3),0,742,0,738,0)
1110 CALL SOUND(Z,A(3),0,442,0,438,0)
1120 CALL SOUND(Z,E,0,661,0,657,0)
1130 CALL SOUND(Z,A(3),0,442,0,438,0)
1140 CALL SOUND(Z,D(3),0,589,0,585,0)
1150 CALL SOUND(Z,E,0,661,0,657,0)
1160 NEXT M
1170 NEXT O
1180 FOR P=1 TO 2
1190 CALL SOUND(Z,D(1),0,D(2),0,D(3),0)
1200 CALL SOUND(Z,D(1),0,D(2),0,D(3),0)
1210 CALL SOUND(Z,A(4),0,882,0,878,0)
1220 CALL SOUND(Z,D(3),0,589,0,585,0)
1230 CALL SOUND(Z,FIS(3),0,742,0,738,0)
1240 CALL SOUND(Z,D(3),0,589,0,585,0)
1250 CALL SOUND(Z,E,0,661,0,657,0)
1260 CALL SOUND(Z,FIS(3),0,742,0,738,0)
1270 NEXT P
1280 FOR Q=1 TO 2
1290 CALL SOUND(Z,H(1),0,H(2),0,D(3),0)
1300 CALL SOUND(Z,H(1),0,H(2),0,D(3),0)
1310 CALL SOUND(Z,H(4),0,990,0,986,0)
1320 CALL SOUND(Z,D(3),0,589,0,585,0)
1330 CALL SOUND(Z,FIS(3),0,742,0,738,0)
1340 CALL SOUND(Z,D(3),0,589,0,585,0)
1350 CALL SOUND(Z,E,0,661,0,657,0)
1360 CALL SOUND(Z,FIS(3),0,742,0,738,0)
1370 NEXT Q
1380 FOR R=1 TO 2
1390 CALL SOUND(Z,G(1),0,G(2),0,D(3),0)
1400 CALL SOUND(Z,G(1),0,G(2),0,D(3),0)
1410 CALL SOUND(Z,H(4),0,990,0,986,0)
1420 CALL SOUND(Z,D(3),0,589,0,585,0)
1430 CALL SOUND(Z,G(3),0,786,0,782,0)
1440 CALL SOUND(Z,D(3),0,589,0,585,0)
1450 CALL SOUND(Z,FIS(3),0,742,0,738,0)
1460 CALL SOUND(Z,G(3),0,786,0,782,0)
1470 NEXT R
1480 FOR S=1 TO 2
1490 CALL SOUND(Z,CIS(1),0,CIS(2),0,CIS(3),0)
1500 CALL SOUND(Z,CIS(1),0,CIS(2),0,CIS(3),0)
1510 CALL SOUND(Z,E,0,661,0,657,0)
1520 CALL SOUND(Z,FIS(3),0,742,0,738,0)
1530 CALL SOUND(Z,B,0,934,0,930,0)
1540 CALL SOUND(Z,FIS(3),0,742,0,738,0)
1550 CALL SOUND(Z,E,0,661,0,657,0)
1560 CALL SOUND(Z,CIS(3),0,556,0,552,0)
1570 NEXT S
1580 GOSUB 1650
1590 RETURN
1600 FOR N=30 TO 0 STEP -.2
1610 CALL SOUND(-100,3500-N*100,0,4000-N*100,0,-5,0)
1620 NEXT N
1630 CALL SOUND(2000,3500,0,4000,0,-5,0)
1640 RETURN
1650 FOR N=0 TO 30 STEP .15
1660 CALL SOUND(-100,3500-N*100,0,4000-N*100,0,-5,0)
1670 NEXT N
1680 RETURN
1690 PRINT "     SPIELBEGINN (-A-)      ":::
1700 FOR N=500 TO 110 STEP -5
1710 CALL SOUND(-100,N,0,N+5,0,N+10,0,-6,0)
1720 NEXT N
1730 CALL KEY(0,K,S)
1740 IF K<>65 THEN 1730
1750 CALL CLEAR
1760 DF=3
1770 ENERGIE=9000
1780 SCORE=0
1790 FEI=0
1800 Q=12
1810 R=0
1820 X=1
1830 Y=1
1840 B=32
1850 CALL SCREEN(2)
1860 CALL COLOR(1,16,1)
1870 CALL COLOR(2,5,1)
1880 CALL COLOR(3,4,1)
1890 CALL COLOR(4,4,1)
1900 CALL COLOR(5,11,1)
1910 CALL COLOR(6,11,1)
1920 CALL COLOR(7,11,1)
1930 CALL COLOR(8,11,1)
1940 FOR N=1 TO 20
1950 CALL HCHAR(INT(RND*23)+1,INT(RND*32)+1,91)
1960 NEXT N
1970 FOR N=1 TO 15
1980 CALL HCHAR(INT(RND*23)+1,INT(RND*32)+1,35)
1990 NEXT N
2000 GOSUB 2800
2010 CALL KEY(0,K,S)
2020 Q=Q-(K=90)+(K=81)
2030 Q=Q-(Q=1)+(Q=23)
2040 CALL HCHAR(Q-1,30,32)
2050 CALL HCHAR(Q+1,30,32)
2060 CALL HCHAR(Q,30,33)
2070 IF ENERGIE<1 THEN 2760
2080 IF FEI=11 THEN 3050
2090 IF K=32 THEN 2490
2100 IF R<>0 THEN 2210
2110 IF RND<.3 THEN 2010
2120 F$="*+<=>"
2130 FEI=FEI+1
2140 F=INT(RND*5)+1
2150 R=ASC(SEG$(F$,F,1))
2160 Y=INT(RND*23)+1
2170 X=1
2180 CALL GCHAR(Y,X,B)
2190 GOSUB 2800
2200 GOTO 2010
2210 CALL HCHAR(Y,X,B)
2220 X=X+1
2230 Y=Y-(RND>.5)+(RND>.5)
2240 Y=Y-(Y=0)+(Y=24)
2250 IF X=33 THEN 2610
2260 CALL GCHAR(Y,X,B)
2270 CALL HCHAR(Y,X,R)
2280 IF B=33 THEN 2950
2290 IF X<27 THEN 2310
2300 GOTO 2010
2310 IF RND<.9 THEN 2010
2320 ZEI=92
2330 IF K<>62 THEN 2350
2340 ZEI=93
2350 CALL SOUND(100,1000,0,1100,0,1200,0)
2360 CALL HCHAR(Y,X+1,ZEI,INT(30-X)/2)
2370 CALL KEY(0,K,B)
2380 IF K=66 THEN 2430
2390 CALL HCHAR(Y,X+1,ZEI,30-X)
```

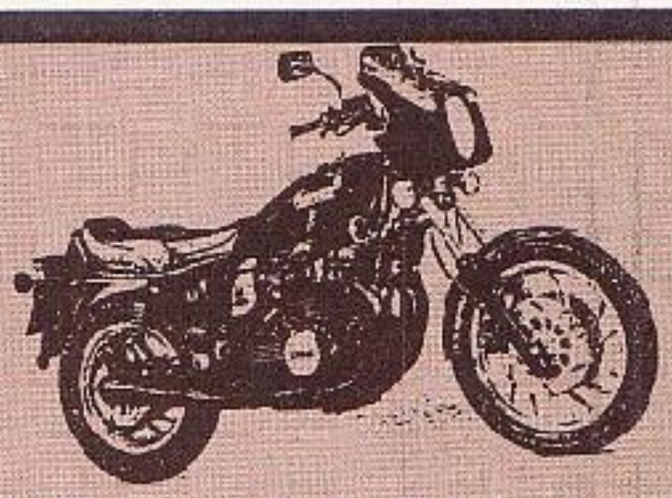
```
2400 CALL HCHAR(Y,X+1,32,30-X)
2410 IF Y=Q THEN 2950
2420 GOTO 2010
2430 CALL HCHAR(Q,29,40)
2440 CALL HCHAR(Y,X+1,ZEI,29-X)
2450 CALL HCHAR(Y,X+1,32,29-X)
2460 CALL HCHAR(Q,29,32)
2470 ENERGIE=ENERGIE-500
2480 GOTO 2010
2490 CALL SOUND(100,1800,8,1810,8,1820,8)
2500 CALL HCHAR(Q,1,95,29)
2510 CALL HCHAR(Q,1,32,29)
2520 ENERGIE=ENERGIE-250
2530 IF Q<>Y THEN 2100
2540 R=0
2550 CALL HCHAR(Y,X,B)
2560 SCORE=SCORE+100
2570 CALL SOUND(100,-5,0)
2580 CALL SOUND(100,-6,0)
2590 CALL SOUND(100,-7,0)
2600 GOTO 2100
2610 A$="          GAME OVER
2620 DF=DF-1
2630 CALL HCHAR(Q,30,32)
2640 IF DF>0 THEN 1800
2650 YS=9
2660 GOSUB 4590
2670 A$="    (HISCORE,END,START)"
2680 YS=11
2690 GOSUB 4590
2700 GOSUB 4430
2710 CALL KEY(0,K,S)
2720 IF K=72 THEN 4090
2730 IF K=83 THEN 1750
2740 IF K<>69 THEN 2710
2750 STOP
2760 ENERGIE=0
2770 GOSUB 2800
2780 A$="          GAME OVER"
2790 GOTO 2650
2800 S$=STR$(SCORE)
2810 E$=STR$(ENERGIE)
2820 S$=SEG$("0000",1,4-LEN(S$))&S$
2830 E$=SEG$("0000",1,4-LEN(E$))&E$
2840 D$=""
2850 FOR N=1 TO DF
2860 D$=D$&" &"
2870 NEXT N
2880 FOR M=N TO 3
2890 D$=D$&"  "
2900 NEXT M
2910 A$="  SCORE"&S$&" ENERGIE"&E$&D$
2920 YS=24
2930 GOSUB 4590
2940 RETURN
2950 CALL HCHAR(Q,30,34)
2960 FOR N=1 TO 20
2970 CALL COLOR(1,10,1)
2980 CALL SOUND(-300,-7,0)
2990 CALL COLOR(1,11,1)
3000 NEXT N
3010 CALL HCHAR(Q,30,32)
3020 CALL HCHAR(Y,X,32)
3030 GOTO 2610
3040 CALL CLEAR
3050 REM
3060 Y=Q
3070 X=30
3080 B=32
3090 DATA 0000000000000FF,000000FF003F70FF,00FF00E031BA7FFF,00FE02C22212FFFF,1F
0000IF3FFFFFFF,FF0E10FFFFFFFFFF
3100 DATA F0000IE3FFFBF9F8,703F00FF00000000,7FBA31E000FF0000,FF1222C202FE0000,FF
FF3F1F00001F00,FFFFFFFF100EFF00
3110 DATA F9FBFFE30100F000
3120 RESTORE 3050
3130 FOR N=110 TO 122
3140 READ Z$
3150 CALL CHAR(N,Z$)
3160 NEXT N
3170 CALL SCREEN(5)
3180 A$="    nopqrst"
```



```
3190 YS=21
3200 GOSUB 4590
3210 A$="      uvwxyz"
3220 YS=22
3230 GOSUB 4590
3240 CALL COLOR(10,2,1)
3250 CALL COLOR(11,2,1)
3260 CALL COLOR(12,2,1)
3270 XD=-3
3280 YD=3
3290 GOSUB 2800
3300 FOR ENERGIE=ENERGIE TO 0 STEP -10
3310 CALL KEY(0,K,S)
3320 XD=XD+(K=68)-(K=67)
3330 YD=YD+(K=83)-(K=90)
3340 CALL HCHAR(Y,X,B)
3350 X=X+SGN(XD)
3360 IF XD>-5 THEN 3380
3370 X=X+SGN(XD)
3380 Y=Y+SGN(YD)
3390 IF YD>-5 THEN 3420
3400 IF YD<5 THEN 3420
3410 Y=Y+SGN(YD)
3420 IF X>1 THEN 3440
3430 X=1
3440 IF X<31 THEN 3460
3450 X=31
3460 IF Y>1 THEN 3480
3470 Y=1
3480 IF Y<24 THEN 3500
3490 Y=23
3500 CALL GCHAR(Y,X,B)
3510 IF ABS((B>113)=(B<117))=1 THEN 3550
3520 CALL HCHAR(Y,X,38)
3530 NEXT ENERGIE
3540 GOTO 4630
3550 IF YD>5 THEN 2950
3560 CALL HCHAR(Y-SGN(YD),X-SGN(XD),38)
3570 F=698
3580 XS=500
3590 Z=150
3600 CALL SOUND(XS,C(3),0,528,0,518,0)
3610 CALL SOUND(XS,G(3),0,789,0,779,0)
3620 CALL SOUND(Z,F,0,703,0,693,0)
3630 CALL SOUND(Z,E,0,664,0,654,0)
3640 CALL SOUND(Z,D(3),0,592,0,582,0)
3650 CALL SOUND(XS,C(4),0,1057,0,1042,0)
3660 CALL SOUND(XS,G(3),0,779,0,789,0)
3670 CALL SOUND(Z,F,0,693,0,703,0)
3680 CALL SOUND(Z,E,0,664,0,654,0)
3690 CALL SOUND(Z,D(3),0,592,0,582,0)
3700 CALL SOUND(XS,C(4),0,1052,0,1042,0)
3710 CALL SOUND(XS,G(3),0,784,0,779,0)
3720 CALL SOUND(Z,F,0,703,0,693,0)
3730 CALL SOUND(Z,E,0,664,0,654,0)
3740 CALL SOUND(Z,F,0,703,0,693,0)
3750 CALL SOUND(XS,D(3),0,592,0,582,0)
3760 FOR ENERGIE=ENERGIE TO 9000 STEP 111
3770 E$=STR$(ENERGIE)
3780 LET E$=SEG$("0000",1,4-LEN(E$))&E$
3790 FOR N=1 TO 4
3800 CALL HCHAR(24,N+19,ASC(SEG$(E$,N,1)))
3810 NEXT N
3820 CALL HCHAR(Y,X,B)
3830 NEXT ENERGIE
3840 CALL HCHAR(20,1,32,10)
3850 FOR Q=Y-1 TO 12 STEP -1
3860 CALL HCHAR(Y,X,B)
3870 CALL GCHAR(Q,X,B)
3880 CALL HCHAR(Q,X,38)
3890 Y=Q
3900 NEXT Q
3910 FOR QW=X TO 30
3920 CALL HCHAR(Y,X,B)
3930 CALL GCHAR(Y,QW,B)
3940 CALL HCHAR(Y,QW,33)
3950 X=QW
3960 NEXT QW
3970 SCORE=SCORE+300
3980 IF SCORE<3000 THEN 4020
3985 IF DF>2 THEN 4020
3990 IF SC1=1 THEN 4020
4000 SC1=1
4010 DF=DF+1
4020 IF SCORE<6000 THEN 4060
4025 IF DF>2 THEN 4060
4030 IF SC1<>1 THEN 4060
4040 SC1=2
4050 DF=DF+1
4060 GOSUB 2800
4070 CALL SCREEN(1)
4080 GOTO 1790
4090 CALL CLEAR
4100 CALL SCREEN(2)
4110 CALL COLOR(2,3,1)
4120 FOR N=3 TO 8
4130 IF N>5 THEN 4160
4140 CALL COLOR(N,9,1)
4150 GOTO 4170
4160 CALL COLOR(N,8,1)
4170 NEXT N
4180 PRINT "H I G H S C O R E - TABELLE":::
4190 FOR N=1 TO 10
4200 N$=STR$(N)&".PLATZ: "
4210 IF N>9 THEN 4230
4220 N$=" "&N$
4230 PRINT N$;HS$(N)
4240 PRINT "          (";HS(N);")"
4250 NEXT N
4260 FOR N=1 TO 2000
4270 CALL KEY(0,K,S)
4280 IF S<>0 THEN 4300
4290 NEXT N
4300 CALL CLEAR
4310 A$="    !HIGHSCORE,END,START)"
4320 YS=11
4330 GOSUB 4590
4340 CALL COLOR(1,16,1)
4350 CALL COLOR(2,5,1)
4360 CALL COLOR(3,4,1)
4370 CALL COLOR(4,4,1)
4380 CALL COLOR(5,11,1)
4390 CALL COLOR(6,11,1)
4400 CALL COLOR(7,11,1)
4410 CALL COLOR(8,11,1)
4420 GOTO 2710
4430 IF SCORE>HS(10)THEN 4450
4440 RETURN
4450 A$="   GEBEN SIE IHREN NAMEN EIN"
4460 YS=14
4470 GOSUB 4590
4480 INPUT N$
4490 FOR N=1 TO 10
4500 IF HS(N)<=SCORE THEN 4520
4510 NEXT N
4520 FOR M=9 TO N STEP -1
4530 HS(M+1)=HS(M)
4540 H$(M+1)=H$(M)
4550 NEXT M
4560 H$(N)=N$
4570 HS(N)=SCORE
4580 RETURN
4590 FOR N2=1 TO LEN(A$)
4600 CALL HCHAR(YS,N2,ASC(SEG$(A$,N2,1)))
4610 NEXT N2
4620 RETURN
4630 FOR N=Y TO 24 STEP 2
4640 CALL HCHAR(Y,X,B)
4650 CALL GCHAR(N,X,B)
4660 CALL HCHAR(N,X,38)
4670 CALL COLOR(1,10,1)
4680 Y=N
4690 NEXT N
4700 A$="    YOUR TANKS ARE EMPTY    "
4710 YS=10
4720 GOSUB 4590
4730 FOR N=1 TO 200
4740 NEXT N
4750 A$="                                "
4760 GOSUB 4590
4770 GOTO 2950
```

SIX-DAYS

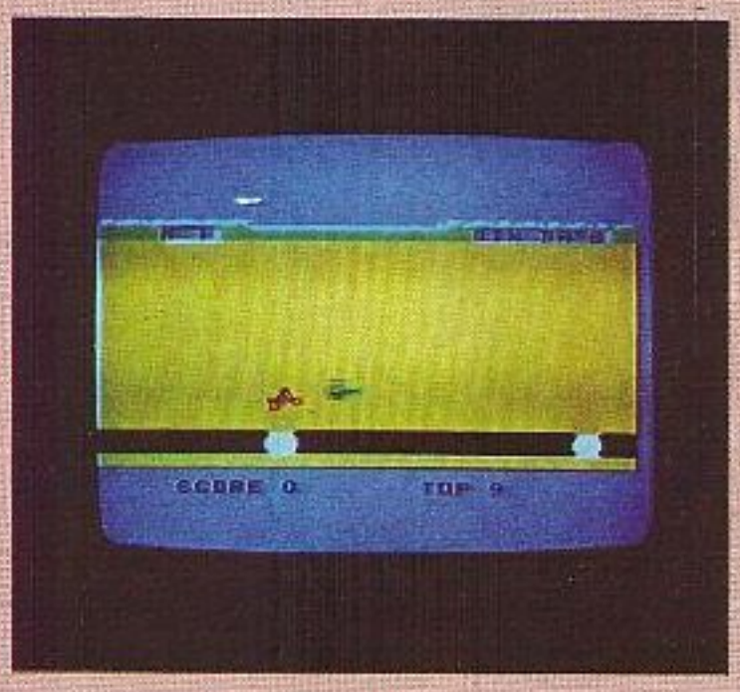
für den TI-99/4A Ext. Basic + Joystick



Ziel des Spieles ist es, einen Zweiradfahrer so sicher über eine bestimmte Wegstrecke zu führen, daß eine vom Programm vorgegebene Höchstzeit nicht überschritten wird. Man sollte nun glauben, dies bedeute keine wesentliche Schwierigkeit, sofern man von der Möglichkeit eines Vollrausches absieht. Weit gefehlt: Unserem tapferen Biker kommen unvermutet Wasserlöcher entgegen, die er wie ein echter Helldriver überspringen muß. Dazu kommt noch ein bösartiger Helikopter im Tiefflug einher, der den geruhsamen Ausflug unseres wackeren Helden mit einem Karambolagerennen zu verwechseln scheint.
Das Spiel besteht aus drei Runden: nach jeder überstandenen Runde erhöht sich das Risiko durch Verlängerung der Wegstrecke und Kürzung der Laufzeit. Mit jeder bewältigten Runde jedoch gewinnen wir auch ein Motorrad. Enden tut unser Spiel, wenn alle Motorräder verbraucht sind. Die Punktzahl wird unter "Score" ausgewertet – die Höchstwertung verbleibt unter "Top"! Hals- und Beinbruch!

Variablenliste
T Top (höchste erreichte Punktzahl. High-Score)
A Score (Punktzahl des Spielers)
M Anzahl der zur Verfügung stehenden Motorräder
L Zunahme der Geschwindigkeit
TI Zeit (laufende Zeit des jeweiligen Spielers)
W derzeit zurückgelegte Wegstrecke
WL Strecke, die zurückgelegt werden muß
TL Zeit (höchstzulässige Zeit, um eine Runde zu bestehen)

Flußdiagramm:
10 - 90 !
100 - Top-Anzeige
110 - 240 Vorspann
250 - 570 Spielaufbau
580 - 720 Spielablauf
730 - 800 Spielende (Neubeg. mit jeder Taste)
810 - 890 Erreichen der nächsten Runde (einschließlich Siegerehrung)




```
 10 ! ********************
 20 ! *   SIX - DAYS     *
 30 ! *                  *
 40 ! * WOLFGANG KLEIN   *
 50 ! * HAFENSTR. 12     *
 60 ! * 2000 WEDEL(H)    *
 70 ! *                  *
 80 ! *   MAI 1984       *
 90 ! ********************
100 T=0
110 CALL CLEAR :: CALL SCREEN(12)
120 CALL MAGNIFY(4)
130 CALL CHAR(91,"7E8199991999817E")
140 CALL CHAR(100,"0103030301030303C7F7F3E455545200CC0C0C0B030D0968BEB74FEF595D
10E");
150 DISPLAY AT(6,8):"S I X - D A Y S"
160 DISPLAY AT(15,4):"PRESS ANY KEY TO START"
170 DISPLAY AT(24,7):"( WOLFGANG KLEIN"
180 CALL SOUND(125,523,0):: CALL SOUND(125,523,0):: CALL SOUND(125,880,0):: CALL
 SOUND(125,783,0):: CALL SOUND(125,698,0)
190 CALL SOUND(125,659,0):: CALL SOUND(125,783,0):: CALL SOUND(125,698,0):: CALL
 SOUND(125,659,0):: CALL SOUND(125,587,0)
200 CALL SOUND(500,523,0):: CALL SOUND(500,440,0)
210 CALL SPRITE(#1,100,2,67,256,0,20)
220 FOR I=1 TO 350 :: NEXT I
230 CALL MOTION(#1,0,0)
240 CALL KEY(0,K,S):: IF S=0 THEN 240 :: CALL DELSPRITE(ALL)
250 CALL CLEAR :: CALL SCREEN(6)
260 CALL MAGNIFY(3)
270 A=0 :: M=3 :: L=0 :: TI=0 :: W=0 :: WL=2000 :: TL=25
280 CALL CHAR(33,"FFFFFFFFFFFFFFFF");! BERGE
290 CALL CHAR(34,"0000000007CFEFFF");! BERGE
300 CALL CHAR(35,"00000000083C7FFFF");! BERGE
310 CALL CHAR(36,"0000000007CFCF1FF");! BERGE
320 CALL CHAR(40,"FFFFFFFFFFFFFFFF");! ERDE
330 CALL CHAR(96,"FFFFFFFFFFFFFFFF");! FAHRBAHN
340 CALL CHAR(104,"0000000000000007BFF3C0000000003000000000000000000E0C0000000000000
000");! FLUGZEUG
350 CALL CHAR(108,"7F3F3F7F7FFFFF7FFFFFFF7F7F3F3F7FFEFCFCFEFEFFFFFFFFFFFFFEFEFCF
```

```
CFE")! WASSER
360 CALL CHAR(112,"3000007F01013F85C5C5C7C3F92523F000000C0000000805FEFEE5800000
080")! RUBSCHRAUBER
370 CALL CHAR(116,"0001030303070707BFFEFDBD4454443800C0C3C020A090B07EB9B5910E0C0
000")! MOTORRAD IM SPRUNG
380 CALL CHAR(120,"404141402412233303A43513B5444443802AC13080C8C14D636FEFAFE151C1
10E")! ZUSAMMENSTOSS
390 CALL CHAR(124,"3F3F1F7F9E9C9A4E2E174F5321030F1FFCFCF3FEF9F9F9F2B4E8F22A84C0F
0F8")
400 CALL COLOR(1,1,1):: CALL COLOR(2,1,1):: CALL COLOR(9,1,1)
410 CALL HCHAR(3,1,34):: CALL HCHAR(3,2,35):: CALL HCHAR(3,3,36):: CALL HCHAR(3,
4,34):: CALL HCHAR(3,5,35)
420 CALL HCHAR(3,6,36):: CALL HCHAR(3,7,34):: CALL HCHAR(3,8,35):: CALL HCHAR(3,
9,36)
430 CALL HCHAR(3,21,34):: CALL HCHAR(3,22,35):: CALL HCHAR(3,23,36):: CALL HCHAR
(3,24,34):: CALL HCHAR(3,25,35)
440 CALL HCHAR(3,26,36):: CALL HCHAR(3,27,34):: CALL HCHAR(3,28,35):: CALL HCHAR
(3,29,36):: CALL HCHAR(3,30,34)
450 CALL HCHAR(3,31,35):: CALL HCHAR(3,32,36)
460 CALL HCHAR(4,1,33,32):: CALL HCHAR(5,1,40,32*15):: CALL HCHAR(20,1,96,32*2):
: CALL HCHAR(22,1,40,32)
470 CALL COLOR(1,4,1):: CALL COLOR(2,12,1):: CALL COLOR(9,2,1)
480 DISPLAY AT(24,4)SIZE(12):"SCORE";A :: DISPLAY AT(24,18)SIZE(12):"TOP";T
490 DISPLAY AT(4,3)SIZE(3):"M";M :: DISPLAY AT(4,21)SIZE(8):"SIX DAYS"
500 CALL SOUND(125,523,0):: CALL SOUND(125,587,0):: CALL SOUND(125,659,0):: CALL
 SOUND(125,587,0):: CALL SOUND(500,523,0)
510 FOR I=1 TO 200 :: NEXT I
520 CALL DELSPRITE(ALL):: CALL MAGNIFY(3)
530 CALL SPRITE(#2,104,14,1,256,0,-5)
540 CALL SPRITE(#1,100,9,138,82)
550 CALL SPRITE(#3,108,15,153,200)
560 CALL SPRITE(#4,112,3,-18,210,0,-30)
570 CALL SPRITE(#5,108,15,153,57)
580 W=W+ABS(L):: IF W>WL THEN 810 :: TI=TI-.1 :: IF TI>=TL THEN 690 :: CALL SOUN
D(-1000,-7,20,110-(L*10),20)
590 CALL COINC(ALL,C):: IF C THEN 690
600 CALL JOYST(1,X,Y)
610 IF X=4 THEN 630 ELSE IF X=-4 THEN 650
620 IF Y=4 THEN 670 ELSE 580
630 L=L-.5 :: CALL SOUND(-1000,-7,20,110-(L*10),20):: CALL MOTION(#5,0,L,#3,0,L)
:: IF L<-120 THEN L=-120
640 GOTO 580
650 L=L+1 :: CALL SOUND(-1000,-7,20,120-(L*10),20):: CALL MOTION(#5,0,L,#3,0,L):
: IF L>=-1 THEN L=-1
660 GOTO 580
670 CALL PATTERN(#1,116):: CALL LOCATE(#1,125,82):: FOR I=1 TO 20 :: CALL COINC(
#1,#4,16,C):: IF C THEN 690
680 NEXT I :: CALL PATTERN(#1,100):: CALL LOCATE(#1,133,82):: GOTO 580
690 CALL PATTERN(#1,120):: CALL SOUND(-1000,-5,0):: CALL MOTION(#3,0,0,#5,0,0)
700 FOR I=1 TO 50 :: NEXT I :: M=M-1 :: DISPLAY AT(4,4)SIZE(2):"";M :: IF M=0 TH
EN 730 :: CALL PATTERN(#1,100)
710 L=0 :: IF TI>=TL THEN TI,W=0
720 GOTO 530
730 CALL DELSPRITE(ALL):: DISPLAY AT(12,7)SIZE(11):"GAME - OVER"
740 A=INT(A+(W/TI)):: DISPLAY AT(24,9)SIZE(7):"";A :: IF A>T THEN T=A :: DISPLAY
 AT(24,22)SIZE(0):"";T
750 CALL SOUND(125,523,0):: CALL SOUND(125,523,0):: CALL SOUND(125,880,0):: CALL
 SOUND(125,783,0):: CALL SOUND(125,698,0)
760 CALL SOUND(125,659,0):: CALL SOUND(125,783,0):: CALL SOUND(125,698,0):: CALL
 SOUND(125,659,0):: CALL SOUND(125,587,0)
770 CALL SOUND(500,523,0):: CALL SOUND(500,440,0)
780 CALL MAGNIFY(4):: CALL SPRITE(#10,100,3,121,256,0,20)
790 CALL KEY(0,K,S):: IF S=0 THEN 790
800 CALL DELSPRITE(ALL):: GOTO 110
810 CALL DELSPRITE(ALL):: CALL MAGNIFY(4):: CALL SPRITE(#1,100,9,120,256,0,10)
820 FOR I=1 TO 300 :: NEXT I :: CALL MOTION(#1,0,0)
830 CALL SPRITE(#28,124,9,120,20*8)
840 WL=WL+1000 :: TL=TL+12 :: A=INT(A+(W/TI)):: DISPLAY AT(24,9)SIZE(8):"";A ::
TI=0 :: W=0
850 CALL SOUND(500,1046,0):: CALL SOUND(125,783,0):: CALL SOUND(125,783,0):: CAL
L SOUND(125,659,0):: CALL SOUND(125,587,0)
860 CALL SOUND(125,783,0):: CALL SOUND(500,1046,0):: CALL SOUND(125,1046,0):: CA
LL SOUND(500,783,0):: CALL SOUND(125,659,0)
870 CALL SOUND(1000,587,0):: FOR I=1 TO 1000 :: NEXT I
880 L=0 :: M=M+1 :: DISPLAY AT(4,4)SIZE(2):"";M
890 GOTO 520
```

Zauberhoehle

für den TI-99 Extended Basic

Sie haben beschlossen, aus einer Höhle ein goldenes, magisches Schwert zu holen, von dem die Leute im Dorf erzählen. Doch wären Sie auch so verwegen, wenn Sie wüßten, daß sich in der Höhle ein Zauberer aufhält, der sich mit Flüchen, Eisentüren, Gruben etc. verteidigt?

Nachdem Sie das Spiel gestartet haben, wird nach dem "letzten Highscore" gefragt. Hat der Computer Ihre Eingabe akzeptiert, beginnt das Spiel.
Die Höhle ist in 20 Abschnitte - oder kürzer "Kammern" - unterteilt, und in der 20. Kammer steht das gesuchte Schwert. In jedem Abschnitt muß man sich vor Eisentüren, Gruben und Bergen oder vor den Flüchen des Zaubereres schützen. Um die Kammern heil zu durchqueren, kann man folgende Aktionen durchführen:
Taste E – Sie machen einen Sprung
Taste S – Sie legen sich hin und rutschen ein Stück
Taste D – Sie werfen eine Bombe
Taste X – Sie können vom Steg auf die darunter liegende Ebene wechseln (Taste I bringt Sie von der Ebene auf den Steg). Diese Aktion nennt sich Tiefensprung (Tiefsp.)
Taste F – Sie können zu Beginn einer Kammer einfach losgehen, wenn Sie nicht gleich eine Aktion (Sprung, Wurf...) durchführen wollen.
Die Gefahren überwinden Sie wie folgt:
a) Eisentür: Sprung oder darunter herkriechen (= Wurf)
b) Grube: Sprung oder gekonnter Hechter (= Wurf)
c) Berg: Sprung oder Bombe (Sprengstoff)
d) Fluch: Schnell den Buchstaben auf der Tastatur drücken, der auf dem Spielfeld angezeigt wird. Der Zauberer stößt 10 Flüche nach Ihnen aus. Haben Sie alle abgewehrt, gilt die Kammer als durchquert. (Der Zauberer kommt, wenn Z in Zeile 410 den Wert 2 hat)
Da Sie vom Schwert angezogen werden, erhöht sich Ihre Geschwindigkeit von Abschnitt zu Abschnitt. Sie müssen immer früher und schneller reagieren. Wenn Sie z.B. springen, wird von der Sprungzahl ein Punkt abgezogen. Ist die Null erreicht, ist kein Sprung mehr möglich. Gleiches gilt bei allen anderen Aktionen.
In jeder Kammer ist ein Schatz. Sie müssen es sich genau überlegen, ob Sie sich ihn holen wollen (Sprung). Natürlich bringt der Schatz Punkte, aber vielleicht brauchen Sie gerade den Sprung zum Überwinden eines Gegenstandes, den Sie gerade zuvor zum Schatzholen verwendet haben...
Der Computer läßt maximal 3 Sprünge und 4 Würfe zu. Haben Sie gerade eine Kammer durchquert, und haben z.B. noch 4 Sprünge gut, dann setzt der Computer die Sprungzahl von 4 auf 3 zurück.
Werden Sie irgendwie verzaubert, so wird Sie 3 mal ein Lebensbringer wiedererwecken.
Und noch etwas: Man kann die Kammer nicht verlassen, wenn man sich auf der Ebene befindet.

Hinweise zum Spiel:
1) Erscheint Ihnen der Fluch zu schnell, dann ändern Sie in Zeile 1570 die Zahl "-20" in eine größere (-19,-18...) ab.
2) Finden Sie die Anzahl der Kammern zu groß, so ist in den Zeilen 470, 520, 1870, 2510 die Zahl "23" in eine kleinere abzuändern.
3) Erreicht man in der 20. Kammer nicht das Schwert, so ist das Spiel zu Ende und der Zauberer hat gesiegt.
4) Haben Sie zu wenig Sprünge oder Würfe, so muß man manchmal einen Lebensbringer opfern, um welche dazu zu erhalten. Den Tiefensprung sollten Sie sich für die letzten Kammern (ab 16.) aufheben.
5) Und natürlich darf man nicht erwarten, daß man das Spiel gewinnt, denn dieses Programm fordert wirklich das Reaktionsvermögen sowie das Glück heraus.

Variablenliste:
P: Aktuelle Punktzahl
HP: Höchste Punktzahl
SP: Sprungzahl
TI: Tiefensprunganzahl
WU: Wurfzahl
KA: Aktuelle Kammer = KA-3
SE: Sprengstoffzahl
L: Lebensbringerzahl
I: Wert für verschiedene For-Next-Schleifen
DEL: Wert für die Verzögerungen
Z: Zufallszahl
CO: Variable bei "Call Coinc"
K: Für die Key-Abfrage
S: Für die Key-Abfrage
X: X-Wert bei "Call Position"

```
30 !*********************************************************
40 !*                                                       *
50 !* ZAUBERHOEHLE  -  fuer den  TI-99/4A+Extended  Basic *
60 !*                                                       *
70 !* (C) by R. Broeschel - Huelsbergstr. 79 - 463 Bochum *
80 !*                                                       *
90 !*********************************************************
100 DISPLAY AT(11,4)ERASE ALL:"DER LETZTE HIGHSCORE" :: DISPLAY AT(13,4):"LAG BE
I      PUNKTEN"
110 ACCEPT AT(13,12)SIZE(4)VALIDATE(DIGIT," "):HP$
120 IF HP$="" THEN 110 ELSE HP=VAL(HP$)
130 RANDOMIZE
140 CALL DELSPRITE(ALL)
150 CALL CLEAR :: CALL SCREEN(14):: CALL MAGNIFY(3)
160 !************************** ZEICHENDEFINITIONEN ***************************
170 CALL CHAR(36,"003001010.03C3318F0303030707070B0B0B0B0B0C0B0C3E0L0C0E0E0E0
E")
180 CALL CHAR(40,RPT$("03",12)&"0000000"&RPT$("0C",12))
190 CALL CHAR(44,"B04B4474222444488"&RPT$("0",19)&"B04B4148E"&RPT$("0",19))
200 CALL CHAR(88,"013426006844402010102146E0024400DC0460084808CAEC6C3A650B1940440
10")
210 CALL CHAR(92,RPT$("0",34)&"B0402020307BFCFC7B3")
220 CALL CHAR(96,"0103030103070F1B.F03030306060L0C06B0L0C080CCE0F0DBLLC0L0L0e03030
18")
230 CALL CHAR(100,"0103030103070F1B1F0303030303030380C0C080C0E0F0D8CCCCC0C0C0C0C
0E")
240 CALL CHAR(104,"11131311:B0F07"&RPT$("03",9)&"88C8C888D8F0E"&RPT$("0C",9))
250 CALL CHAR(108,RPT$("0",24)&"012377=F"&RPT$("0",24)&"C1E1FFFF")
260 CALL CHAR(112,RPT$("0",16)&"18:B3C183C7E7E3C"&RPT$("0",17))
270 CALL CHAR(116,RPT$("0",18)&"3CJE3D3D3E3C7E"&RPT$("0",17))
280 CALL CHAR(120,RPT$("F",16),121,"FFFF7F5F585A5212",122,"FFFFFF7E7E3C0C00")
290 CALL CHAR(128,"FFFF")
300 CALL CHAR(136,RPT$("F",60))
310 CALL CHAR(140,"0101010303070307OF1F3F1F3F3FFFFF8080B0C0C0E0E0FBF0F0F8F8FCFCF
EFF")
320 SC,VE,ZA,P=0 :: WU,SE=1 :: TI=2 :: SP,L=3 :: KA=4
330 !************************** SPIELFELD AUFBAUEN ****************************
340 CALL CLEAR :: CALL DELSPRITE(ALL)
350 FOR I=1 TO 14 :: CALL COLOR(I,L,1):: NEXT I
360 DISPLAY AT(1,1):"PUNKTE         HIGH         SPRUNG   TIEFSP     WURF    KAM
MER    SPRENGSTOFF       LEBENSBRINGER"
370 FOR I=5 TO 9 :: CALL HCHAR(I,3,120,28):: NEXT I :: FOR I=23 TO 24 :: CALL HC
HAR(I,3,120,28):: NEXT I
380 CALL HCHAR(13,3,128,28):: CALL HCHAR(5,3,99,28):: CALL VCHAR(1,1,99,48):: CA
LL VCHAR(1,31,99,48)
390 DISPLAY AT(9,1):"zyyyzyzzyzyzyyzzyzyzyzzyzyzy"
400 IF KA<5 OR KA>21 THEN 430
410 Z=INT(RND*5)+1 :: IF Z=2 THEN 1490
420 !************************** SPRITES SETZEN ********************************
430 FOR I=1 TO 2 :: CALL SPRITE(#I+5,136,14,136,INT(RND*180)+30):: NEXT I :: CAL
L COINC(#6,#7,30,CO1):: IF CO1=-1 THEN 430
440 Z=INT(RND*2)+1 :: IF Z<>1 THEN ZCH=140 :: SPR=5 ELSE ZCH=40 :: SPR=9
450 CALL SPRITE(#SPR,ZCH,14,121,INT(RND*180)+30)
460 CALL COINC(#SPR,#6,30,CO1):: CALL COINC(#SPR,#7,30,CO2):: IF CO1=-1 OR CO2=-
1 THEN 450
470 IF KA=23 THEN 500
480 Z=INT(RND*2)+1 :: IF Z=1 THEN HO=90 ELSE HO=160
490 CALL SPRITE(#10,INT((RND*2)+28)*4,14,HO,INT(RND*180)+30)
500 CALL SPRITE(#8,136,14,160,225)
510 GOSUB 1440
520 IF KA=23 THEN GOSUB 2800 :: CALL LOCATE(#15,12)
530 GOSUB 2880
540 CALL SPRITE(#1,100,8,121,15)
550 DISPLAY AT(11,2):"BITTE EINE TASTE DRUECKEN"
560 CALL KEY(0,K,S):: IF S=0 OR K<>70 AND K<>69 AND K<>88 AND K<>68 AND K<>83 TH
EN 560
570 DISPLAY AT(11,2)
580 GOTO 610
590 !************************** HAUPTPROGRAMM *********************************
600 CALL KEY(0,K,S):: IF S=0 THEN 550
610 IF K=69 THEN 690
620 IF K=88 THEN 770
630 IF K=68 THEN 1070
640 IF K=83 THEN 1230
650 CALL MOTION(#1,0,KA):: CALL PATTERN(#1,96):: GOSUB 1810 :: FOR DEL=1 TO 10-K
A :: NEXT DEL
660 CALL PATTERN(#1,100):: GOSUB 1810 :: FOR DEL=1 TO 10-KA :: NEXT DEL
670 GOTO 600
680 !************************** SPRUNG ****************************************
690 IF SP<1 THEN 650 ELSE SP=SP-1 :: GOSUB 1370
700 CALL SOUND(100,1500,0)
710 FOR I=-18 TO 18 STEP 4 :: CALL MOTION(#1,I,6)
720 CALL COINC(#1,#10,20,CO3):: IF CO3=-1 THEN GOSUB 1850
730 GOSUB 1810 :: NEXT I
740 CALL MOTION(#1,0,0):: CALL POSITION(#1,X,Y):: CALL LOCATE(#1,121,Y)
750 GOSUB 1810 :: GOTO 600
760 !************************** STEG-EBENE-WECHSEL ****************************
770 IF TI<1 THEN 650 ELSE TI=TI-1 :: GOSUB 1300
780 CALL MOTION(#1,0,0):: GOSUB 1810
790 CALL SOUND(400,500,0,-5,0)
800 CALL PATTERN(#1,104)
810 CALL POSITION(#1,X,Y):: CALL LOCATE(#1,129,Y)
820 FOR DEL=1 TO 100 :: NEXT DEL
830 CALL MOTION(#1,2,0)
840 FOR DEL=1 TO 97 :: NEXT DEL
850 CALL MOTION(#1,0,0)
860 CALL LOCATE(#1,161,Y)
870 FOR DEL=1 TO 100 :: NEXT DEL
880 CALL PATTERN(#1,100):: CALL SOUND(100,1234,0)
890 FOR DEL=1 TO 10-KA :: NEXT DEL
900 CALL MOTION(#1,0,KA)
910 CALL PATTERN(#1,96)
920 CALL KEY(0,K,S)
930 IF K=73 THEN 970
940 CALL COINC(#1,#8,10,CCC):: IF CCC=-1 THEN GOSUB 1850
```

Y: Y-Wert bei "Call Position"
VE: Läßt nur zu, daß man 1 mal den Highscore bekommt
AN: Bewirkt, daß, wenn die Bombe über den Bildrand hinaus kommt, die Eisentür trifft oder nichts trifft, nur die Bombe explodiert
TR: Bestimmt, wie lang die Bombe fliegt
KY: Ist der ASCII-Code für das Zeichen, das beim Zauberer erscheint
ZE: Gibt die Zeile an, in der das Zeichen ASC(KY) angezeigt wird
SPA: Gibt die Spalte an, in der...
WI: 4-malige Wiederholung der Töne, die bei verlorenem Spiel kommen
LA: Lautstärke (0-29) der Töne...
SC: Wert des Schatzes mal 20
ZCH: Zeichen für Berg oder Eisentür
SPR: Spritenummer für Berg oder Eisentür
ZA: Indikator, ob in der durchquerten Kammer der Zauberer war

Definition der Zeichen und Belegung der Sprites:

#1 das Männchen (Farbe 8)
#2 der Zauberer (Farbe 5)
#3 die Fluchwellen (Farbe 13)
#4 die Bombe (Farbe 7)
#5 der Berg (Farbe 7)
#6 Falle (=Grube) (Farbe 2)
#7 Falle (Farbe 2)
#8 Falle (Farbe 2) versperrt den Ausgang der Ebene
#9 Eisentür (Farbe 5)
#10 Schatz bzw. das Schwert (Farbe 16 bzw. 12)

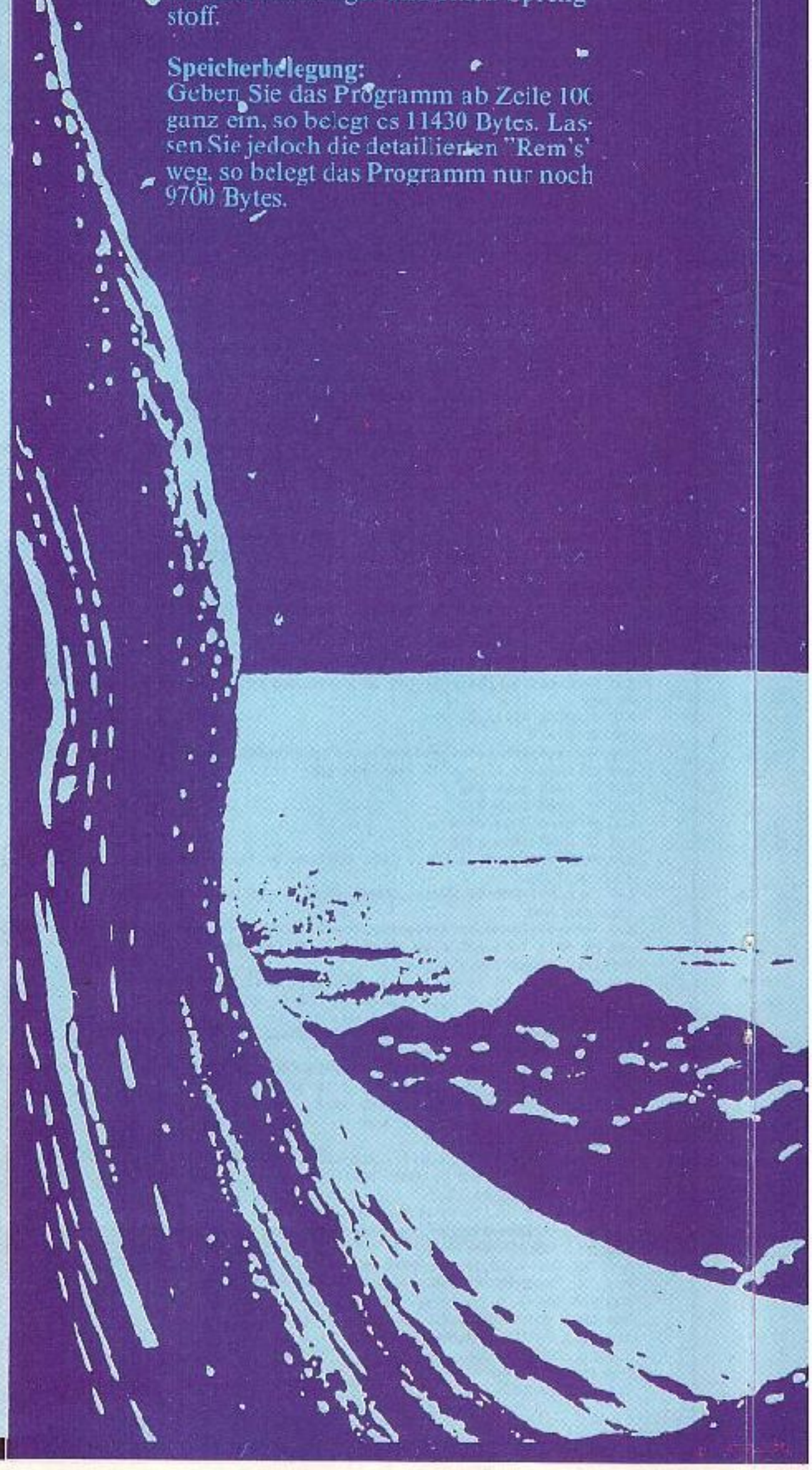
Zeichen:
36 - Zauberer
40 - Eisentür
44 - Fluchwellen
88 - Explosion bzw. Verzauberung des Männchens
92 - Bombe
96 - gehendes Männchen
100 - stehendes Männchen
104 - Männchen beim Tiefensprung
108 - Männchen beim Wurf
112 - Zepter
116 - Becher
120 - Höhlenboden und Teil der Decke
121 + 122 - Höhlenzapfen
128 - Steg
136 - Falle (=Grube)
140 - Berg

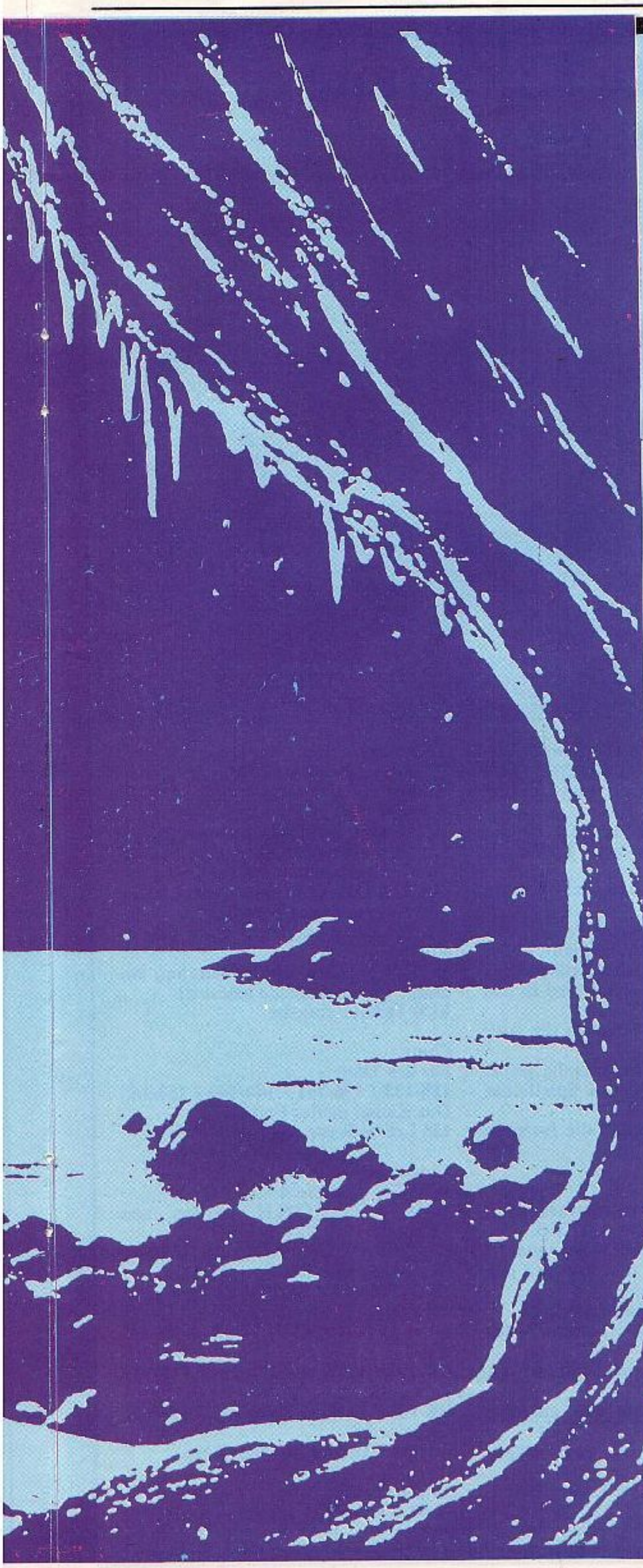
Punktezählung:
Hat man einen Abschnitt durchquert, so erhält man (KA+1)mal 10 Punkte. Für einen geholten Schatz erhalten Sie zwischen 100 und 200 Punkte.
Haben Sie es geschafft, die Kammer mit dem Zauberer zu durchqueren, so gibt es dafür zwischen 100 und 300 Punkte

```
1980 CALL SOUND(500,659,0,261,0,329,0)
1990 CALL SOUND(250,523,0)
2000 CALL SOUND(250,659,0)
2010 CALL SOUND(500,783,0,261,0,329,0)
2020 CALL SOUND(250,659,0)
2030 CALL SOUND(250,783,0)
2040 CALL SOUND(500*I,1046,0,261,0,329,0)
2050 NEXT I
2060 FOR DEL=1 TO 400 :: NEXT DEL
2070 CALL SCREEN(2):: CALL CHARSET
2080 FOR I=8 TO 14 :: CALL COLOR(I,2,2):: NEXT I
2090 FOR I=1 TO 8 :: CALL COLOR(I,16,2):: NEXT I
2100 RESTORE 2110 :: FOR I=8 TO 20 STEP 2 :: READ A$ :: DISPLAY AT(I,1):A$ :: NE
XT I
2110 DATA "SUPER ! ! !","DU HAST ES GESCHAFFT DEN","ZAUBERER DAS SCHWERT ZU","EN
TREISSEN!","","DU BIST DER NEUE  HERRSCHER"
2120 DATA "UEBER DIE ZAUBERHOEHLE!!!"
2130 DISPLAY AT(24,1):"EIN NEUES SPIEL? (J/N)"
2140 CALL KEY(0,K,S)
2150 IF K=78 THEN 2910
2160 IF K=74 THEN CALL COLOR(1,14,14):: GOTO 130
2170 GOTO 2140
2180 !******************** EXPLOSION DES MAENNCHENS ***************************
2190 CALL MOTION(#1,0,0):: CALL DELSPRITE(#3):: CALL PATTERN(#1,88
2200 FOR I=0 TO 30 :: CALL SOUND(-1000,-7,I):: NEXT I
2210 FOR DEL=1 TO 500 :: NEXT DEL
2220 IF VE=0 THEN GOSUB 2710
2230 L=L-1 :: IF L<0 THEN 2290
2240 SP=SP+2 :: WU=WU+2
2250 SE=SE+1
2260 IF P>=HP THEN HP=P
2270 GOTO 340
2280 !************************** SPIEL VERLOREN ***************************
2290 FOR WI=1 TO 4
2300 FOR LA=0 TO 29
2310 CALL SOUND(-100,710-20*LA,LA)
2320 NEXT LA
2330 NEXT WI
2340 CALL SOUND(500,261,0)
2350 CALL SOUND(300,195,0)
2360 CALL SOUND(500,207,0)
2370 CALL SOUND(300,174,0)
2380 CALL SOUND(500,195,0)
2390 CALL SOUND(300,155,0)
2400 CALL SOUND(150,146,0)
2410 CALL SOUND(250,155,0)
2420 CALL SOUND(1200,130,0)
2430 CALL DELSPRITE(ALL):: FOR I=1 TO 14 :: CALL COLOR(I,2,2):: NEXT I :: CALL S
CREEN(2):: CALL CHARSET
2440 FOR I=1 TO 8 :: CALL COLOR(I,16,2):: NEXT I
2450 RESTORE 2460 :: FOR I=7 TO 21 STEP 2 :: READ A$ :: DISPLAY AT(I,1):A$ :: NE
XT I
2460 DATA "LEIDER HAST DU ES NICHT","GESCHAFFT DEM ZAUBERER","DAS MAGISCHE SCHWE
RT","ZU ENTREISSEN."
2470 DATA "DEM ZAUBERER DAGEGEN IST","ES GELUNGEN SEINE","HERRSCHAFT UEBER DIE H
OEHLE","ZU VERLAENGERN..."
2480 GOTO 2130
2490 !****************** ROUTINE NACH DURCHQUERUNG EINER KAMMER ****************
2500 CALL DELSPRITE(#1)
2510 IF KA>=23 THEN KA=KA-1 :: L=0 :: GOTO 2290
2520 KA=KA+1 :: SP=SP+1 :: WU=WU+1 :: IF ZA=1 THEN 2560
2530 FOR I=1 TO KA :: P=P+10 :: GOSUB 1350 :: CALL SOUND(-1,1000,0):: NEXT I
2540 FOR DEL=1 TO 200 :: NEXT DEL
2550 FOR I=1 TO SC :: P=P+20 :: GOSUB 1350 :: CALL SOUND(1,1000,0,2000,0,3000,0)
:: NEXT I
2560 SC=0 :: ZA=0
2570 IF SP>3 THEN SP=3
2580 IF WU>4 THEN WU=4
2590 IF KA=12 OR KA=20 THEN 2610 ELSE 2660
2600 !********************************** BONUS ***********************************
2610 FOR I=1 TO KA*2
2620 DISPLAY AT(12,13):"BONUS"
2630 P=P+20 :: CALL SOUND(1,1000,0,1500,0,2000,0):: DISPLAY AT(12,13)
2640 GOSUB 1350 :: NEXT I :: TI=TI-1 :: SE=SE+1
2650 GOSUB 2680
2660 IF VE=0 THEN GOSUB 2710
2670 FOR DEL=1 TO 400 :: NEXT DEL
2680 IF P>=HP THEN HP=P
2690 GOTO 340
2700 !****************************** HIGHSCORE ********************************
2710 IF P<HP THEN RETURN
2720 DISPLAY AT(12,4):"DU HAST DEN HIGHSCORE" :: L=L+1 :: VE=1
2730 CALL SOUND(2000,523,0,784,0,1047,1,-1,0)
2740 GOSUB 2680
2750 FOR DEL=1 TO 500 :: NEXT DEL
2760 HP=P :: GOSUB 1360 :: GOSUB 1420 :: CALL SOUND(2000,131,0,392,0,1047,0,-1,0
)
2770 FOR DEL=1 TO 500 :: NEXT DEL
2780 RETURN
2790 !******************** ROUTINE VOR DER  20. KAMMER ************************
2800 CALL CHAR(112,RPT$("01",16)&"0907060D0701010;"&RPT$("80",16)&"909090E0808
")
2810 FOR I=5 TO 10
2820 CALL POSITION(#1,X,Y)
2830 IF Y=200 THEN CALL DELSPRITE(#1)
2840 NEXT I
2850 CALL SPRITE(#10,112,17,121,210)
2860 RETURN
2870 !************************** ANZEIGEROUTINE ***************************
2880 FOR I=1 TO 8 :: ON I GOSUB 1350,1360,1370,1380,1390,1400,1410,1420 :: NEXT
I
2890 RETURN
2900 !****************************** ENDE *********************************
2910 CALL CLEAR :: DISPLAY AT(12,10):"TSCHUESS!!"
2920 FOR DEL=1 TO 1500 :: NEXT DEL
2930 CALL CLEAR
```

Besonderheiten:
Überwinden Sie einen Abschnitt, so erhalten Sie einen Sprung und einen Wurf dazu.
Gelingt es Ihnen nicht, eine Kammer zu überwinden, so bekommen Sie einen Sprengstoff, zwei Sprünge und zwei Würfe hinzu.
Bei den Kammern 8 und 16 gibt es einen Bonus; d.h. Sie erhalten 320 bzw. 640 Punkte, einen Tiefensprung und einen Sprengstoff dazu.
Ist Ihre Punktzahl höher oder gleich der des Highscores, so gibt es dafür einen Lebensbringer und einen Sprengstoff.

Speicherbelegung:
Geben Sie das Programm ab Zeile 100 ganz ein, so belegt es 11430 Bytes. Lassen Sie jedoch die detaillierten "Rem's" weg, so belegt das Programm nur noch 9700 Bytes.

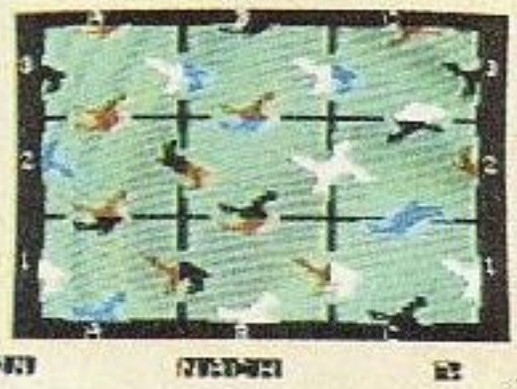


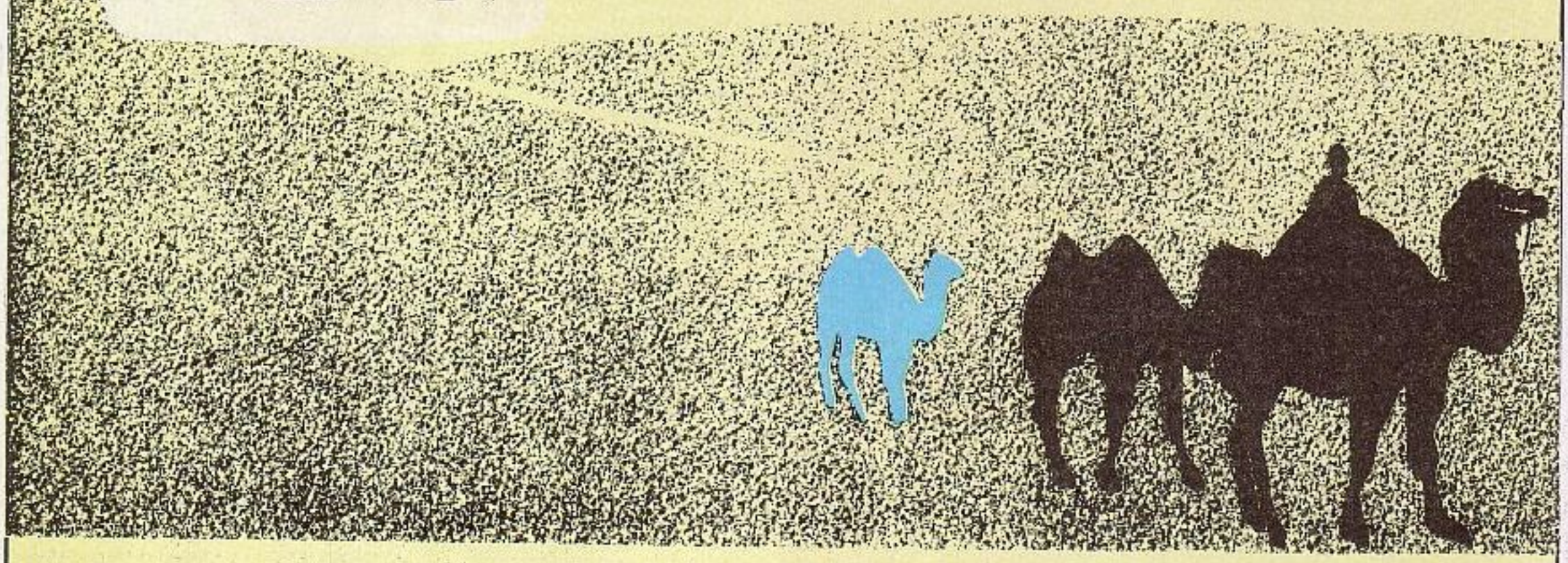


```
950 GOSUB 1810
960 GOTO 500
970 CALL MOTION(#1,0,0)
980 CALL PATTERN(#1,104):: CALL SOUND(100,1400,0)
990 CALL MOTION(#1,-12,0)
1000 FOR DEL=1 TO 99 :: NEXT DEL
1010 CALL MOTION(#1,0,3):: GOSUB 1810
1020 CALL POSITION #1,X,Y:: CALL LOCATE(#1,121,Y)
1030 CALL SOUND(100,1500,0,131,0)
1040 CALL PATTERN(#1,100)
1050 GOSUB 1810 :: GOTO 600
1060 !********** SPRENGSTOFF **********
1070 IF SC<1 THEN 450 ELSE SC=SC-1 :: GOSUB 1410
1080 CALL MOTION(#1,0,3)
1090 CALL POSITION(#1,X,Y):: CALL SPRITE(#4,92,7,121,Y+10,0,12)
1100 CALL COINC(#4,#5,10,CO5)
1110 CALL COINC(#4,#9,10,CO6)
1120 IF CO6=-1 THEN 1160
1130 IF CO5=-1 THEN 1170
1140 CALL POSITION(#4,X,Y):: IF Y>220 THEN 1160
1150 TR=TR+1 :: IF TR<7 THEN 1100
1160 CALL PATTERN(#4,08):: CALL MOTION(#4,0,0):: AN=1 :: GOTO 1180
1170 CALL MOTION(#4,0,0):: CALL PATTERN(#4,68,#5,88)
1180 FOR I=0 TO 30 :: CALL SOUND(-1000,-7,I):: NEXT I
1190 IF AN=0 THEN CALL DELSPRITE(#4,#5)
1200 CALL DELSPRITE(#4)
1210 AN=0 :: TR=0 :: GOSUB 1810 :: GOTO 500
1220 !********** WURF **********
1230 IF WU<1 THEN 450 ELSE WU=WU-1 :: GOSUB 1390
1240 CALL MOTION(#1,0,0)
1250 CALL SOUND( 100,1000,0,2000,0)
1260 CALL PATTERN(#1,108)
1270 CALL MOTION(#1,0,10)
1280 FOR I=1 TO 15
1290 CALL COINC(#1,#5,3,COA):: IF COA=-1 THEN 2190
1300 NEXT I
1310 CALL POSITION(#1,X,Y):: IF Y>220 THEN 2500
1320 CALL MOTION(#1,0,0):: CALL PATTERN(#1,100)
1330 GOSUB 1810 :: GOTO 600
1340 !********** ANZEIGE **********
1350 DISPLAY AT(1,7)SIZE(6):P :: RETURN
1360 DISPLAY AT(1,20)SIZE(7):HP :: RETURN
1370 DISPLAY AT(2,7)SIZE(3):SP :: RETURN
1380 DISPLAY AT(2,17)SIZE(3):TI :: RETURN
1390 DISPLAY AT(2,25)SIZE(3):WU :: RETURN
1400 DISPLAY AT(3,7)SIZE(3):KA-3 :: RETURN
1410 DISPLAY AT(3,22)SIZE(3):SE :: RETURN
1420 DISPLAY AT(4,15)SIZE(2):L :: RETURN
1430 !********** ERZEUGEN DER FARBEN **********
1440 CALL COLOR(9,14,14,12,11,2,1,8,2,13,7,2,3,16,2,4,16,2)
1450 FOR I=5 TO 8 :: CALL COLOR(I,8,2):: NEXT I
1460 CALL COLOR(#4,2,#7,2,#5,7):: CALL COLOR(#8,2,#9,5,#10,16)
1470 RETURN
1480 !********** DER ZAUBERER **********
1490 CALL VCHAR(1,1,99,48):: CALL VCHAR(1,31,95,48)
1500 GOSUB 2880 :: GOSUB 1440
1510 FOR I=5 TO 7 :: CALL COLOR(I,16,2):: NEXT I
1520 CALL CHAR(36,"00C001010103C3310F03030307070703808080808080C030C0C0C0C0E3E0E
0E0");
1530 CALL CHAR(44,"80F044424222444488"&RPT$("0",19)&"804844488"&RPT$("0",19))
1540 CALL SPRITE(#1,100,8,121,30
1550 CALL SPRITE(#2,36,5,121,210)
1560 FOR I=1 TO 10
1570 CALL SPRITE(#3,44,13,123,194,0,-20)
1580 CALL SOUND(400,-5,0)
1590 KY=INT(RND*23)+65
1600 ZE=INT(RND*8)+10
1610 SPA=INT(RND*19)+6
1620 CALL HCHAR(ZE,SPA,KY)
1630 CALL KEY(0,K,S)
1640 IF K=KY THEN 1660
1650 CALL COINC(ALL,CO4):: IF CO4=-1 THEN CALL HCHAR(ZE,SPA,32):: GOTO 2190 ELSE
 1630
1660 CALL DELSPRITE(#3)
1670 CALL HCHAR(ZE,SPA,32)
1680 CALL SOUND(100,500,0,1500,0 -1,0)
1690 NEXT I
1700 FOR I=30 TO 0 STEP -1
1710 CALL SOUND(-1000,-7,I)
1720 NEXT I
1730 CALL DELSPRITE(#2)
1740 FOR I=0 TO 30
1750 CALL SOUND(-400,-7,I)
1760 NEXT I
1770 FOR I=1 TO INT(RND*30)+10
1780 CALL SOUND(10,1000,0):: P=P+10 :: GOSUB 1350 :: NEXT I
1790 ZA=1 :: GOTO 2510
1800 !********** PRUEF-ROUTINE **********
1810 CALL POSITION(#1,X,Y):: IF Y>215 THEN 2500
1820 CALL COINC(#1,#10,20,CO7):: IF CO7=-1 THEN GOSUB 1850
1830 CALL COINC(ALL,CO8):: IF CO8=-1 THEN 2190
1840 RETURN
1850 SC=INT(RND*5)+5 :: CALL DELSPRITE(#10)
1860 CALL SOUND(10,1000,0)
1870 IF KA=23 THEN 1900
1880 RETURN
1890 !********** SPIEL GEWONNEN **********
1900 CALL MOTION(#1,0,0)
1910 CALL LOCATE(#1,121,196)
1920 CALL SPRITE(#10,112,12,113,202)
1930 CALL DELSPRITE(#4,#5,#6,#7,#9)
1940 CALL PATTERN #1,104)
1950 FOR I=1 TO 2
1960 CALL SOUND(250,391,0)
1970 CALL SOUND(250,523,0)
```

Die verrückte Karawane

für den TI-99/4A





Als Vorlage diente dem Programm "Die verrückte Karawane" das bekannte Puzzlespiel "Die verflixte Hexe".
Neun quadratische Teilstücke sind so in einem Quadrat zu legen, daß die auf den Teilstücken befindlichen Köpfe und Schwänze der Kamele farblich übereinstimmen. Alles weitere kann den Spielregeln des Programmes, geschrieben in TI-Basic, entnommen werden.

Programmaufbau:
120-980: Titel und Spielanleitung
1000-1030: Farben
1040-1350: Charanweisungen
Das Kamel in den vier Richtungen wird für jeweils alle vier Farben definiert.
1360-1510: Der Rahmen des Spielfeldes wird dargestellt
1520-1590: Charanweisungen für den Rahmen der Teilstücke
1620-1730: Die neun Teilstücke werden in das Spielfeld gezeichnet. Die Verteilung ist zufällig
1750-1820: Das Eingabefeld wird dargestellt
1830-1880: Löschen der alten Koordinaten und der alten Drehung mit Sound
1890-2050: Eingabe der neuen Koordinaten und Drehung
2060-2130: Bestimmung der Feldwerte und der linken oberen Spalten- und Zeilennummer der beiden eingegebenen Teilstücke
2140-2190: Die beiden Teilstücke werden vom Spielfeld entfernt
2200-2380: Bestimmung der Teilstücknummern und deren Positionen, Verzweigung zu den jeweiligen Teilstücken. (Teilstück 4 und 6 sind gleich)
2390-4060: Verzweigung zu der jeweiligen Position des Teilstücks. Ermittlung von O,R,U und L
4070-4900: Bestimmung der x-Werte in den vier Positionen
4920-4960: Feste x-Werte
(x(1-36) sind die ASCII-Zeichencodes für jedes Rasterfeld eines Teilstücks)
4970-5260: Das Teilstück wird in das Spielfeld gezeichnet
5270-5560: Überprüfung, ob die Aufgabe gelöst ist
5570-5620: Wenn die Aufgabe gelöst ist, wird das Umfeld des Spielfeldes umgeändert
5630-5790: Anzeige, daß die Aufgabe erfüllt ist

Variablenliste:
A,E,F,I,J,N,R,T,V Schleifenvariable
T1$,T2$,T3$ String für den Titel
F$ String für die Farbanweisungen
R1$,R2$ Strings für den Rahmen des Spielfeldes
EING$ String für die Eingabezeile
TST (2,2) Ergibt die Nummer des Teilstücks
TSTP(2,2) Ergibt die Position des Teilstücks
V1,Vs Feldwerte des bei "von" eingegebenen Teilstücks
N1,N2 Feldwerte des bei "nach" eingegebenen Teilstücks
NR,NR1 Nummer des 1. und des 2. Teilstücks
P,P1 Position des 1. und des 2. Teilstücks
D Eingegebene Drehung
Z Zeile
S Spalte
Z1,S1 Für die Zwischenspeicherung von Anfangszeile und -spalte des bei "von" eingegebenen Teilstücks
O,R,U,L Das Kamel oben, recht, unten und links auf dem Teilstück
X1-X36 Die Rasterfelder eines Teilstücks
Y(23),B Werden zur Berechnung benötigt, ob die Aufgabe gelöst ist
K(5) Tastatureingabe

Chars
Für den Titel:
128-133 / 136-141 / 144-149 / 152-157 Das Kamel in vier Farben
135 Leeres Rasterfeld

Für das Spiel:
39 Volles Rasterfeld
40-46 A,B,C,V,O,N,H werden neu definiert
47 Pfeil
55 Leeres Rasterfeld
56-61 / 62-67 / 68-73 / 74-79 Kamel schwarz
80-84 / 86-91 / 92-97 / 98-103 Kamel blau
104-109 / 110-115 / 116-121 / 122-127 Kamel rot
128-133 / 134-139 / 140-145 / 146-151 Kamel weiß
152-159 Rahmen der Teilstücke

Aufgabe gelöst:
48-53 B,R,A,V,O,! werden neu definiert

```
120 CALL CLEAR
130 DIM TST(2,2),TSTP(2,2),Y
(23),K(5)
140 CALL SCREEN(12)
150 FOR I=13 TO 16
160 CALL COLOR(1,12,12)
170 NEXT I
180 CALL CHAR(135,"")
190 S=8
200 FOR I=128 TO 152 STEP 8
210 RESTORE 1050
220 FOR J=I TO I+5
230 READ J$
240 CALL CHAR(J,J$)
250 NEXT J
260 CALL HCHAR(12,S,1)
270 CALL HCHAR(12,S+1,I+1)
280 CALL HCHAR(12,S+2,I+2)
290 CALL HCHAR(12,S+3,135)
300 CALL HCHAR(13,S,135)
310 CALL HCHAR(13,S+1,I+3)
320 CALL HCHAR(13,S+2,I+4)
330 CALL HCHAR(13,S+3,135)
340 CALL HCHAR(14,S,135)
350 CALL HCHAR(14,S+1,I+5,2)
360 CALL HCHAR(14,S+3,135)
370 S=S+5
380 NEXT I
390 T1$="DIE VERRUECKTE"
400 T2$="KARAWANE"
410 FOR T=1 TO 14
420 CALL HCHAR(3,T+9,ASC(SEG
$(T1$,T,1)))
430 NEXT T
440 FOR T=1 TO 8
450 CALL HCHAR(6,T+12,ASC(SE
G$(T2$,T,1)))
460 NEXT T
470 CALL SOUND(300,523,0)
480 CALL SOUND(200,440,0)
490 CALL SOUND(200,371,0)
500 CALL SOUND(150,329,0)
510 CALL SOUND(150,261,0)
520 CALL SOUND(150,220,0)
530 CALL SOUND(250,261,0)
540 CALL SOUND(250,293,0)
550 CALL SOUND(250,440,1)
560 CALL SOUND(350,523,1)
570 CALL SOUND(300,440,2)
580 CALL SOUND(200,391,1)
590 CALL COLOR(13,2,4)
600 CALL COLOR(14,5,4)
610 CALL COLOR(15,7,4)
620 CALL COLOR(16,16,4)
630 T3$="SPIELREGELN ?  (J/N
)"
640 FOR T=1 TO 20
650 CALL HCHAR(22,T+6,ASC(SE
G$(T3$,T,1)))
660 NEXT T
670 CALL KEY(0,KEY,ST)
680 IF ST=0 THEN 670
690 CALL CLEAR
700 CALL CHAR(47,"0078848495
8E84")
710 IF KEY<>74 THEN 990
720 PRINT ::"        H A L L
 O !":::"  TEX LAEDT SIE  ZU
 EINEM"::"  KAMELRITT DU
RCH DIE"::"  WUESTE EIN.":::
730 PRINT "  DOCH LEIDER IST
 DIE KA-"::"  RAWANE ETWAS D
URCHEINAN-"::"  DER GERA
TEN."::
740 PRINT "  BEVOR SIE ENDLI
CH STAR-"::"  TEN KANN,IST E
RST EINE"::" AUFGABE ZU
 ERFUELLEN."::TAB(25);"-->"
750 CALL KEY(0,KEY,ST)
760 IF ST=0 THEN 750
770 CALL CLEAR
780 PRINT "  DIE KAMELE BEFI
NDEN SICH"::"  AUF EINER QUA
DRATISCHEN"::"  FLAECHE,
DIE WIEDERUM AUS"::
790 PRINT "  NEUN (EBENFALLS
 QUADRA-"::"  TISCHEN)  TEIL
STUECKEN"::"  BESTEHT.":
:
800 PRINT "  DIESE SIND SO Z
U LEGEN,"::"  DASS DER KOPF
UND DER"::"  SCHWANZ  EI
NES  KAMELES"::
810 PRINT "  GLEICHFARBIG SI
ND."::"  ES GIBT ZWEI LOESUN
GEN."::TAB(25);"-->"
820 CALL KEY(0,KEY,ST)
830 IF ST=0 THEN 820
840 CALL CLEAR
850 PRINT "  BEWEGUNG DER TE
ILSTUECKE"::"  -VON-"::'  EI
NGABE DER KOORDINATEN"::
"  (ZB.A1) FUER DAS ZU BE-':
:
860 PRINT "  WEGENDE TEILSTU
CCK."::"  -NACH-"::"  EINGAB
E DER KOORDINATEN"::"  F
UER DIE NEUE STELLE DES"::
870 PRINT "  TEILSTUECKS. DA
S HIER"::"  BEFINDLICHE  TEI
LSTUECK"::"  TAUSCHT MIT
 DEM ERSTEN"::
880 PRINT "  DEN PLATZ.";TAB
(25);"-->"
890 CALL KEY(0,KEY,ST)
900 IF ST=0 THEN 890
910 CALL CLEAR
920 PRINT "  -/-"::"  GEWUEN
SCHTE DREHUNG IM"::"  UHRZEI
GERSINN DES BEI"::"  -VO
N- EINGEGEBENEN TEIL-"::
930 PRINT "  STUECKS.DAS ZWE
ITE TEIL-"::"  STUECK WIRD N
ICHT GEDREHT."":::
940 PRINT "  0 = KEINE DREHU
NG"::"  1 = DREHUNG UM  90 G
RAD"::
950 PRINT "  2 = DREHUNG UM
180 GRAD"::"  3 = DREHUNG UM
 270 GRAD"::"     VIEL E
RFOLG !";TAB(25);"-->"
960 CALL KEY(0,KEY,ST)
970 IF ST=0 THEN 960
980 CALL CLEAR
990 REM  FARBEN,CHARANWEISU
NGEN,RAHMEN SPIELFELD
1000 F$="ADLBPBCPBDBDEBDFBDG
EDHEDIEDJGDKGDLGDMPDNPDOPDPB
D"
1010 FOR F=0 TO 45 STEP 3
1020 CALL COLOR(ASC(SEG$(F$,
1+F,1))-64,ASC(SEG$(F$,2+F,1
))-64,ASC(SEG$(F$,3+F,1)
)-64)
1030 NEXT F
1040 DATA 0000000018187878,C
0C0D0D0D3C0C0E,E0E0C08,01010
107FFFF1F1F,1F0F0F3FFFFF
020C,00000B0F0407
1050 DATA 0000000F0F0303,000
000C00001C7FF,000000000C0E0
F0,FF1F1F8F0F0C0C0C,FCFE
FDFD9C1C0C0C,14141C3
1060 DATA 1E1E181B,070303010
1010303,0000000001030707,FBF
BFFFF0A0A08,3040FFFFFCF
FFFF8,0000E020F01
1070 DATA 00C0C0F0F0,FFE38,0
F0703,303030F0F0FBF0FF,3C303
839BFBF7F3F,000000000C38
2B28
1080 FOR I=56 TO 128 STEP 24
1090 RESTORE 1040
1100 FOR J=I TO I+5
1110 READ J$
1120 CALL CHAR(J,J$)
1130 NEXT J
1140 NEXT I
1150 FOR I=62 TO 134 STEP 24
1160 RESTORE 1050
1170 FOR J=I TO I+5
1180 READ J$
1190 CALL CHAR(J,J$)
1200 NEXT J
1210 NEXT I
1220 FOR I=68 TO 140 STEP 24
1230 RESTORE 1060
1240 FOR J=I TO I+5
1250 READ J$
1260 CALL CHAR(J,J$)
1270 NEXT J
1280 NEXT I
1290 FOR I=74 TO 146 STEP 24
1300 RESTORE 1070
1310 FOR J=I TO I+5
1320 READ J$
1330 CALL CHAR(J,J$)
1340 NEXT J
1350 NEXT I
1360 CALL CHAR(40,"003844447
C444444")
1370 CALL CHAR(41,"007824243
8242478")
1380 CALL CHAR(42,"003844404
0404438")
1390 CALL CHAR(55,"")
1400 R1$="777(77777)77777*77
77"
1410 R2$="7737777727777171777
"
1420 FOR Z=2 TO 21 STEP 19
1430 FOR R=1 TO 20
1440 CALL HCHAR(Z,R+6,ASC(SE
G$(R1$,R,1)))
1450 NEXT R
1460 NEXT I
1470 FOR S=7 TO 26 STEP 19
1480 FOR R=1 TO 18
1490 CALL VCHAR(R+2,S,ASC(SE
G$(R2$,R,1)))
1500 NEXT R
1510 NEXT S
1520 CALL CHAR(152,"FF808080
8080808")
1530 CALL CHAR(153,"FF010101
01010101")
1540 CALL CHAR(154,"01010101
```

```
010101FF")
1550 CALL CHAR(155,"80808080
8080808FF")
1560 CALL CHAR(156,"FF")
1570 CALL CHAR(157,"01010101
01010101")
1580 CALL CHAR(158,"00000000
000000FF")
1590 CALL CHAR(159,"80808080
8080808")
1600 CALL CHAR(39,"FFFFFFFFF
FFFFFFF")
1610 REM  VERTEILUNG DER TEI
LSTUECKE
1620 RANDOMIZE
1630 FOR NR=1 TO 9
1640 V1=INT(3*RND)
1650 V2=INT(3*RND)
1660 Z=15-5*V1
1670 S=8+6*V2
1680 IF TST(V1,V2)<>0 THEN 1
640
1690 P=INT(4*RND)
1700 TST(V1,V2)=NR
1710 TSTP(V1,V2)=P
1720 ON NR GOSUB 2390,2600,2
810,3020,3230,3020,3440,3650
,3860
1730 NEXT NR
1740 REM   BEWEGUNG DER TEIL
-STUECKE
1750 CALL CHAR(43,"004444444
444281")
1760 CALL CHAR(44,"007C44444
444447C")
1770 CALL CHAR(45,"004484845
44C4C44")
1780 CALL CHAR(46,"004444447
C444444")
1790 EING$="+,-      -(*.
   /"
1800 FOR E=1 TO 20
1810 CALL HCHAR(23,E+4,ASC(S
EG$(EING$,E,1)))
1820 NEXT E
1830 CALL HCHAR(23,10,32,2)
1840 CALL HCHAR(23,20,32,2)
1850 CALL HCHAR(23,27,32,1)
1860 FOR N=330 TO 990 STEP 1
10
1870 CALL SOUND(50,N,1)
1880 NEXT N
1890 FOR I=1 TO 3 STEP 2
1900 CALL KEY(0,K(I),ST)
1910 IF ST=0 THEN 1900
1920 IF K(I)<65 THEN 1900
1930 IF K(I)>67 THEN 1900
1940 CALL HCHAR(23,5*I+5,K(I
)-25)
1950 CALL KEY(0,K(I+1),ST)
1960 IF ST=0 THEN 1950
1970 IF K(I+1)<49 THEN 1950
1980 IF K(I+1)>51 THEN 1950
1990 CALL HCHAR(23,5*I+6,K(I
+1))
2000 NEXT I
2010 CALL KEY(0,K(5),ST)
2020 IF ST=-1 THEN 2010
2030 IF K(5)<48 THEN 2010
2040 IF K(5)>51 THEN 2010
2050 CALL HCHAR(23,27,K(5))
2060 V1=K(2)-49
2070 V2=K(1)-65
2080 N1=K(4)-49
2090 N2=K(3)-65
2100 Z=15-6*N1
2110 S=8+6*N2
2120 Z1=15-6*V1
2130 S1=8+6*V2
2140 FOR I=0 TO 5
2150 CALL HCHAR(Z+I,S,32,6)
2160 NEXT I
2170 FOR I=0 TO 5
2180 CALL HCHAR(Z1+I,S1,32,6
)
2190 NEXT I
2200 NR=TST(V1,V2)
2210 NR1=TST(N1,N2)
2220 D=K(5)-40
2230 P=TSTP(V1,V2)
2240 P1=TSTP(N1,N2)
2250 TST(V1,V2)=NR1
2260 TSTP(V1,V2)=P1
2270 TST(N1,N2)=NR
2280 P=P+D
2290 IF P<4 THEN 2310
2300 P=P-4
2310 TSTP(N1,N2)=P
2320 ON NR GOSUB 2390,2600,2
810,3020,3230,3020,3440,3650
,3860
2330 IF NR=NR1 THEN 1830
2340 P=P1
2350 Z=Z1
2360 S=S1
2370 ON NR1 GOSUB 2390,2600,
2810,3020,3230,3020,3440,365
0,3860
2380 GOTO 1830
2390 ON P-1 GOTO 2400,2450,2
500,2550
2400 D=56
2410 R=134
2420 U=104
2430 L=86
2440 GOTO 4030
2450 D=80
2460 R=74
2470 U=128
2480 L=122
2490 GOTO 4290
2500 D=116
2510 R=98
2520 U=68
2530 L=146
2540 GOTO 4500
2550 D=140
2560 R=110
2570 U=92
2580 L=62
2590 GOTO 4710
2600 ON P+1 GOTO 2610,2660,2
710,2760
2610 D=80
2620 R=110
2630 U=56
2640 L=134
2650 GOTO 4080
2660 D=128
2670 R=98
2680 U=104
2690 L=74
2700 GOTO 4290
2710 D=68
2720 R=146
2730 U=92
2740 L=122
2750 GOTO 4500
2760 D=116
2770 R=62
2780 U=140
2790 L=86
2800 GOTO 4710
2810 ON P+1 GOTO 2820,2870,2
920,2970
2820 D=56
2830 R=110
2840 U=80
2850 L=116
2860 GOTO 4080
2870 D=104
2880 R=74
2890 U=104
2900 L=98
2910 GOTO 4290
2920 D=92
2930 R=122
2940 U=68
2950 L=122
2960 GOTO 4500
2970 D=116
2980 R=86
2990 U=116
3000 L=62
3010 GOTO 4710
3020 ON P+1 GOTO 3030,3080,3
130,3180
3030 D=80
3040 R=134
3050 U=56
3060 L=110
3070 GOTO 4080
3080 D=104
3090 R=98
3100 U=128
3110 L=74
3120 GOTO 4290
3130 D=68
3140 R=122
3150 U=92
3160 L=146
3170 GOTO 4500
3180 D=140
3190 R=62
3200 U=116
3210 L=86
3220 GOTO 4710
3230 ON P+1 GOTO 3240,3290,3
340,3390
3240 D=80
3250 R=134
3260 U=56
3270 L=134
3280 GOTO 4080
3290 D=128
3300 R=98
3310 U=128
3320 L=74
3330 GOTO 4290
3340 D=68
3350 R=146
3360 U=92
3370 L=146
3380 GOTO 4500
3390 D=140
3400 R=62
3410 U=140
3420 L=86
3430 GOTO 4710
3440 ON P+1 GOTO 3450,3500,3
550,3600
3450 D=104
3460 R=134
3470 U=104
3480 L=62
3490 GOTO 4080
3500 D=56
3510 R=122
3520 U=128
3530 L=122
3540 GOTO 4290
3550 D=116
3560 R=74
3570 U=116
```

```
3580 L=146
3590 GOTO 4500
3600 O=140
3610 R=110
3620 U=68
3630 L=110
3640 GOTO 4710
3650 ON P+1 GOTO 3660,3710,3
760,3810
3660 U=56
3670 R=110
3680 U=80
3690 L=134
3700 GOTO 4080
3710 O=120
3720 R=74
3730 U=104
3740 L=98
3750 GOTO 4290
3760 O=92
3770 R=146
3780 U=68
3790 L=122
3800 GOTO 4500
3810 O=116
3820 R=86
3830 U=140
3840 L=62
3850 GOTO 4710
3860 ON P+1 GOTO 3870,3920,3
970,4020
3870 O=104
3880 R=110
3890 U=120
3900 L=62
3910 GOTO 4080
3920 O=56
3930 R=122
3940 U=104
3950 L=146
3960 GOTO 4290
3970 O=140
3980 R=74
3990 U=116
4000 L=122
4010 GOTO 4500
4020 O=116
4030 R=134
4040 U=68
4050 L=110
4060 GOTO 4710
4070 REM  POSITIONEN DER
 TEILSTJECKE
4080 X2=O+5
4090 X3=O+4
4100 X4=O+2
4110 X5=156
4120 X7=159
4130 X9=39
4140 X12=157
4150 X13=L+2
4160 X17=R
4170 X18=R+1
4180 X19=L+4
4190 X20=39
4200 X24=R+3
4210 X25=L+5
4220 X28=U
4230 X30=R+5
4240 X32=U+5
4250 X33=U+3
4260 X34=U+1
4270 X35=158
4280 GOTO 4920
4290 X2=O+5
4300 X3=O+4
4310 X4=O-2
4320 X5=156
4330 X7=L-5
4340 X9=39
4350 X12=R+5
4360 X13=L+3
4370 X17=39
4380 X18=R+4
4390 X19=L+1
4400 X20=L
4410 X24=R+2
4420 X25=159
4430 X28=U
4440 X30=157
4450 X32=U+5
4460 X33=U+3
4470 X34=U+1
4480 X35=150
4490 GOTO 4920
4500 X2=156
4510 X3=O+1
4520 X4=O+3
4530 X5=O+5
4540 X7=L+5
4550 X9=O
4560 X12=R-5
4570 X13=L-3
4580 X17=39
4590 X18=R-4
4600 X19=L+1
4610 X20=L
4620 X24=R+2
4630 X25=159
4640 X28=39
4650 X30=157
4660 X32=158
4670 X33=U+2
4680 X34=U+4
4690 X35=U+5
4700 GOTO 4920
4710 X2=156
4720 X3=O+1
4730 X4=O+3
4740 X5=O+5
4750 X7=159
4760 X9=O
4770 X12=157
4780 X13=L+2
4790 X17=R
4800 X18=R+1
4810 X19=L+4
4820 X20=39
4830 X24=R+3
4840 X25=L+5
4850 X28=39
4860 X30=R+5
4870 X32=158
4880 X33=U+2
4890 X34=U+4
4900 X35=U+5
4910 REM   ZEICHNEN DER TEIL
-STUECKE
4920 X1=152
4930 X6=153
4940 X8=39
4950 X31=155
4960 X36=154
4970 CALL HCHAR(Z,S,X1)
4980 CALL HCHAR(Z,S+1,X2)
4990 CALL HCHAR(Z,S+2,X3)
5000 CALL HCHAR(Z,S+3,X4)
5010 CALL HCHAR(Z,S+4,X5)
5020 CALL HCHAR(Z,S+5,X6)
5030 CALL HCHAR(Z+1,S,X7)
5040 CALL HCHAR(Z+1,S+1,X8)
5050 CALL HCHAR(Z+1,S+2,X9)
5060 CALL HCHAR(Z+1,S+3,X8,2
)
5070 CALL HCHAR(Z+1,S+5,X12)
5080 CALL HCHAR(Z+2,S,X13)
5090 CALL HCHAR(Z+2,S+1,X8,3
)
5100 CALL HCHAR(Z+2,S+4,X17)
5110 CALL HCHAR(Z+2,S+5,X18)
5120 CALL HCHAR(Z+3,S,X19)
5130 CALL HCHAR(Z+3,S+1,X20)
5140 CALL HCHAR(Z+3,S+2,X8,3
)
5150 CALL HCHAR(Z+3,S+5,X24)
5160 CALL HCHAR(Z+4,S,X25)
5170 CALL HCHAR(Z+4,S+1,X8,2
)
5180 CALL HCHAR(Z+4,S+3,X28)
5190 CALL HCHAR(Z+4,S+4,X8)
5200 CALL HCHAR(Z+4,S+5,X30)
5210 CALL HCHAR(Z+5,S,X31)
5220 CALL HCHAR(Z+5,S+1,X32)
5230 CALL HCHAR(Z+5,S+2,X33)
5240 CALL HCHAR(Z+5,S+3,X34)
5250 CALL HCHAR(Z+5,S+4,X35)
5260 CALL HCHAR(Z+5,S+5,X36)
5270 REM  ABFRAGE ERFOLGT?
5280 CALL GCHAR(5,13,Y(0))
5290 CALL GCHAR(5,14,Y(1))
5300 CALL GCHAR(5,19,Y(2))
5310 CALL GCHAR(5,20,Y(3))
5320 CALL GCHAR(11,13,Y(4))
5330 CALL GCHAR(11,14,Y(5))
5340 CALL GCHAR(11,19,Y(6))
5350 CALL GCHAR(11,20,Y(7))
5360 CALL GCHAR(17,13,Y(8))
5370 CALL GCHAR(17,14,Y(9))
5380 CALL GCHAR(17,19,Y(10))
5390 CALL GCHAR(17,20,Y(11))
5400 CALL GCHAR(8,10,Y(12))
5410 CALL GCHAR(9,10,Y(13))
5420 CALL GCHAR(14,10,Y(14))
5430 CALL GCHAR(15,10,Y(15))
5440 CALL GCHAR(8,16,Y(16))
5450 CALL GCHAR(9,16,Y(17))
5460 CALL GCHAR(14,16,Y(18))
5470 CALL GCHAR(15,16,Y(19))
5480 CALL GCHAR(8,22,Y(20))
5490 CALL GCHAR(9,22,Y(21))
5500 CALL GCHAR(14,22,Y(22))
5510 CALL GCHAR(15,22,Y(23))
5520 FOR A=0 TO 22 STEP 2
5530 B=ABS(Y(A)-Y(A+1))
5540 IF B=1 THEN 5560
5550 RETURN
5560 NEXT A
5570 CALL HCHAR(2,10,55,13)
5580 CALL HCHAR(21,10,55,13)
5590 CALL VCHAR(3,7,55,13)
5600 CALL VCHAR(5,26,55,13)
5610 CALL HCHAR(23,5,32,23)
5620 CALL SCREEN(7)
5630 CALL CHAR(48,"007824243
8242478")
5640 CALL CHAR(49,"007844447
E504844")
5650 CALL CHAR(50,"003844447
C444444")
5660 CALL CHAR(51,"004444444
444281")
5670 CALL CHAR(52,"007C44444
4444478")
5680 CALL CHAR(53,"00101010
010001")
5690 CALL SOUND(350,440,0)
5700 CALL SOUND(175,523,0)
5710 CALL SOUND(250,659,0)
5720 CALL SOUND(400,523,1)
5730 CALL SOUND(300,440,3)
5740 FOR S=0 TO 5
5750 CALL HCHAR(23,11+2*S,48
+S)
5760 NEXT S
5770 FOR V=0 TO 400
5780 NEXT V
5790 GOTO 5610
```

GHOST MANOR

Ghost Manor ist ein geradezu klassisches Action Adventure mit toller Graphik, hoher Geschwindigkeit, vielen Levels und all dem andern, was ein gutes Spiel erfolgreich macht.
In Schwierigkeitsstufe 1 geht es freilich noch recht gemütlich zu: In etwa 20 Sekunden müssen Sie sich Ihren Munitionsvorrat verdienen, um die nächsten Bilder durchstehen zu können. Die Munition erhalten Sie, indem Sie soviel Gespenster als irgend möglich fangen, d.h. das auf dem Bildschirm herumgeisternde Spukwesen mit Ihrer Spielfigur zur Deckung bringen.
In Stufe 2 befinden Sie sich vor Draculas Burg, wo sich alleroris eigenartige Wesen tummeln, die die Burg beschützen.
Um in das Schloß gelangen zu können, müssen Sie Ihre in Level 1 verdienten Pfeile gegen die Monster einsetzen.
Der einzige, der Ihnen wirklich dabei gefährlich werden kann, ist der schwarze Henker, der Sie mit seiner Axt erschlagen will. Ihn können Sie nämlich nicht bezwingen, da er über einen Schutzmantel verfügt. Sobald Sie alle Monster abgeschossen haben, kommen Sie in die Burg selbst, wo Sie sich zunächst im Raum der beweglichen Wände wiederfinden werden. Dort haben Sie die Aufgabe, ein unsichtbares Kreuz aufzustöbern. Sobald Sie es haben, werden die Wände weiß, und Sie gelangen durch den gestrichelten Ausgang in Level 4. Dabei laufen Sie allerdings Gefahr, an die Wände zu stoßen und eines raschen, wenn auch schmerzlosen Todes zu sterben. Der Tod wird für Sie zum Neubeginn: Sie dürfen in Level 1 von vorn beginnen!
Sollte Ihnen jedoch wider Erwarten der Übergang in's vierte Level gelingen, so wird die Lage nicht angenehmer, denn Sie befinden sich in Draculas Gruft, wo der fürstliche Vampir eine Frau in seiner Gewalt hat, die Sie als Kavalier der alten Schule natürlich retten wollen. Das ist jedoch leichter gesagt als getan, da sich Graf Dracula diesbezüglich recht störrisch und wenig kompromißbereit zeigt. Sobald er zudringlich wird, müssen Sie versuchen, irgendwie an ihm vorbei zu kommen, ohne ihn zu berühren. Klappt es, so starten Sie in's nächste Level – andernfalls ist das Ihr Ende... ob Sie jemals erfahren werden, was sich in Level 5 und danach abspielt?

für den TI-99/4A Extended Basic

```
100 ON WARNING NEXT
110 BWR=0
120 CALL DELSPRITE(ALL):: CA
LL CHARSET :: CALL CLEAR
130 PUNKTE=100
140 TRE=0 :: CALL CHAR(67,"3
C4299A1A199423C")
150 U=0
160 CALL SCREEN(5):: DIM F(5
):: F(1)=8 :: F(2)=9 :: F(3)
=12 :: F(4)=4 :: F(5)=14
170 CALL CHAR(97,"FFFFFFFFFF
FFFFFF",98,"0103070F1F3F7FFF
",99,"80C0E0F0F8FCFEFF")
180 CALL CHAR(100,"FFFEFCF8F
0E0C080",101,"FF7F3F1F0F0703
01",134,"FFFFFEFEFCFCF8F
8",135,"F0F0E0E0C0C08080")
190 CALL CHAR(136,"FFFF7F7F3
F3F1F1F",137,"0F0F0707030301
01")
200 PRINT : : : : : :
210 PRINT "  baad b  c baac
baad baaad"
220 PRINT "  a    a  a acca
ad     a     a    acba a  a
ac     a     a ac aaaa a
  a aaar   a     a aa adea a
  a   ea   a  "
230 PRINT "  acba a  a acba
  ba   a     eaad e  d eaad
baad   a  "
240 PRINT : : :
250 PRINT " c   b baac c  b
baac baaac  ac ba adea ac a
adea ad ea  acada acba a
eca a  a ac ba  a   a aaaa a
 ea a  a aaaad"
260 PRINT " a   a a  a a  a
acba a  ac  e   d e  d e  d
eaad e   ec"
270 PRINT : : :
280 FOR I=2 TO 12 :: CALL CO
LOR(I,8,5):: NEXT I
290 DISPLAY AT(22,4):"C BY A
NDREAS FOERSTER"
300 DISPLAY AT(24,3):'PRESS
ANY KEY TO BEGINN '
310 FOR I=2 TO 20
320 CALL KEY(0,K,T):: IF T<>
0 THEN 390
330 CALL SOLOR(09,F(INT(RND*
5)+1),5,14,F(INT(RND*5)+1),5
)
340 CALL SOUND(-500,I*100,I,
1200-I*100,20-I*1)
350 NEXT I
360 FOR I=10 TO 2 STEP -1 ::
 CALL KEY(0,K,T):: IF T<>0 T
HEN 390
370 CALL SOUND(-500,1200-I*1
00,I,I*100,20-I)
380 NEXT I :: GOTO 310
390 W=0 :: CALL CLEAR
400 DISPLAY AT(04,10):"GOST
MANOR" :: DISPLAY AT(18,15):
"   C BY ANDREAS FOERSTE
R"
410 DISPLAY AT(08,08):"1.ANF
AENGER" :: DISPLAY AT(10,08)
:"2.MEISTER" :: DISPLAY
AT(12,08):"3.WELTMEISTER"
420 DISPLAY AT(20,15):'
  TEL.46058/2442"
430 DISPLAY AT(22,8):"6466 G
RUENDAU 5 HAINSTR.18  '
440 CALL COLOR(9,5,5)
450 DISPLAY AT(15,6)BEEP:'ST
UFE (1-3) ?" :: ACCEPT AT(15
,20)VALIDATE("123")SIZE(
-1):GRAD
460 DIM GR2(3):: GR2(1)=5 ::
 GR2(2)=10 :: GR2(3)=15
470 FUNKT1=GRAD
480 GESCH=GR2(GRAD)
490 HACK=GRAD*.3
500 DIM DR1(3):: DR1(1)=2 ::
 DR1(2)=6 :: DR1(3)=8
510 DT=DR1(GRAD)
520 DIM POS1(2):: POS1(1)=-1
0 :: POS1(2)=10
530 GRAD=INT(GRAD*12)
540 Z=131
```

```
550 CALL SCREEN(5):: CALL CO
LOR(1,5,5,9.5,5,14.5,5)
560 Y=150
570 X=160
580 XDIR=-GRAD
590 E=1
600 CALL CLEAR
610 CALL CHAR(40,"FFFFFFFFFF
FFCFFF",41,"FFFFFFFF00000000
")
620 CALL CHAR(42,"99FF7E3C3D
3F3333",43,"0000SAFFE7C7C7C7
",44,"99FF7E3CBCFCCCCC")
630 CALL SPRITE(#3,42,5,17,1
05,#4,43,5,17,121,#5,44,5,17
,137)
640 CALL COLOR(10,5,5,2,5,5)
650 PRINT "((((((((((((((((
(((((((((((((((((((((((((((
((((((((((((((((((((((((
(((((((((((((('
660 PRINT "((((((((((((((((((
(((((((((((((((((((((((((((
((((((((((((((((((((((((
((((((((((((((("
670 PRINT "((((((((((
))(((((((((()))))
    ))))(((("
680 PRINT : : :
690 CALL CHAR(129,"3C7E0D99F
FFFFF55",130,"1310FF0DBD3C24
24")
700 CALL CHAR(131,"3C7EB899F
FFFFFAA")
710 CALL CHAR(97,"3C7EFFFFFF
FFFFFF",98,"0000000000F1F3F3F
",99,"3F3F3F3F3F3F3F3F",
100,"0000000F0FBFCFC",101,"
FCFCFCFCFCFCFCFCFC")
720 CALL CHAR(102,"1F1F3F3F7
F7FFFFF",103,"FFFFFFFFFFFFFF
FF",104,"FBFBFCFCFEFEFFFF
FFFFF")
730 CALL MAGNIFY(2)
740 PRINT "                bd"
750 PRINT "  '          a  ce
a        "
760 PRINT "          bd    fh
 bd                ce    gg
 cp

                    "
770 PRINT "      fggh    fggh
  fggh          gggg   gggg
  gggg          gggg   g
ggg   gggg             gggg   g
ggg   gggg             "
780 PRINT
790 CALL SCREEN(2):: CALL CO
LOR(2,5,7,1,9,9,9,7,9,10,7,7
,14,3,9)
800 CALL COLOR(#3,12,#4,12,#
5,12)
810 CALL SPRITE(#1,130,2,120
,120)
820 CALL SPRITE(#2,129,8,Y,X
,0,XDIR)
830 FOR I=1 TO 220
840 CALL POSITION(#2,B,A)::
IF A<30 OR A>170 THEN 890
850 CALL JOYST(2,X1,Y1):: CA
LL MOTION(#1,-Y1*E,X1*8)
860 CALL COINC(#1,#2,12,P)::
 IF P=-1 THEN U=U+.5 :: CALL
 SOUND(-100,1500,0)
870 NEXT I
880 GOTO 1020
890 REM
900 E=E+1 :: IF E>3 THEN E=1
910 ON E GOSUB 960,980,1000
920 IF Z=131 THEN Z=129 ELSE
 Z=131
930 CALL SPRITE(#2,Z,8,Y,X,0
,XDIR)
940 GOTO 870
950 GOTO 950
960 Y=150 :: X=160 :: XDIR=-
GRAD
970 RETURN
980 Y=110 :: X=80 :: XDIR=GR
AD
990 RETURN
1000 Y=070 :: X=160 :: XDIR=
-GRAD
1010 RETURN
1020 CALL DELSPRITE(ALL):: C
ALL CLEAR
1030 CALL CHAR(120,"01010101
0101010101630000000000000000
0000000300000000000002000
000")
1040 CALL MAGNIFY(3):: CALL
CHAR(40,"1919FFBD3D3E2525250
0000000000000000000002000
0000000000000000000000")
1050 CALL SCREEN(2):: CALL C
OLOR(1,2,2,13,2,2,14,2,2,5,2
,2,6,2,02,7,2,2)
1060 CALL CHAR(83,"FFFFFFFFF
FFFFFFFFF")
1070 CALL CHAR(ASC("H"),"FF1
010FFFFB0B0FF",82,"3C7EFFFFF
FFFFFFF")
1080 PRINT : : : : : : : : :
1090 PRINT "R R R
        R R RHHHHH
        HHHHHHHHH
                HHHHHH
                 HHH"
1100 PRINT "HHH
          HHHHHH
          HHHHHH R R R R
R R R R R R RHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHH"
1110 PRINT "HHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHHHH
HHHHHHHHHHHHHHHHHHHHHHHHH
HHSSHHHHHHHHHHHHHHHHHHHHHHHHH
HSSSSHHHHHHHHHHHH"
1120 PRINT "HHHHHHHHHHHHHSSSS
SSHHHHHHHHHHHHHHHHHHHHHHSSSS
FSHHHHHHHHHHHHHHHHHHHHHHHHH
SSSSSSHHHHHHHHHHHHHHHHHHHHHHHH
SSSSSSHHHHHHHHHHH'
1130 DISPLAY AT(24,1):"AAAAA
AAAAAAAAAAAAAAAAAAAAAAAAA"
1140 CALL SCREEN(2):: CALL C
OLOR(1,6,6,13,7,10,14,2.5,5,
13,13,6,7,10,7,2,5)
1150 CALL SPRITE(#1,40,12,18
5,120)
1160 CALL CHAR(112,"1819FFDC
DCDC3C3C0000000000000000000
0000000000100000000000000
000")
1170 CALL SPRITE(#2,112,2,IN
T(13*8)+2,120,0,10)
1180 CALL CHAR(108,"00000000
01010F0B0B030206000000C00000
1C1C9090F020C0C0406C0000
0000")
1190 CALL SPRITE(#11,108,16,
175,140,0,-10)
1200 CALL CHAR(124,"00000000
09090F03030302060000000000C00
00000080F0FAFCFCFCF4F4F0707
0000")
1210 CALL CHAR(100,"00000080
403C03C1000000030000000000000
00010220C0B00000000000000
0000")
1220 CALL CHAR(104,"00000000
80324381810000000000000000000
0000007CC8181000000000000
000")
1230 CALL SPRITE(#4,104,10,2
0,100,0,9,#5,104,15,30,200,0
,-10,#6,104,2,50,160,0,1
0)
1240 CALL CHAR(116,"3C7E5AFF
663C24180000000000000000000
00000000000000000000000
00")
1250 CALL SPRITE(#7,116,16,1
40,180,0,-5,#8,116,12,160,40
,0,10)
1260 CALL SPRITE(#03,1,6,02,
120,110,0,5)
1270 CALL SPRITE(#9,104,8,09
0,10,0,-10)
1280 CALL JOYST(2,X,Y):: CAL
L MOTION(#1,0,X*4)
1290 DISPLAY AT(1,1):"  ":ST
R$(INT(U))
1300 A9=INT(3+RND*3)+1 :: CA
LL PATTERN(#11,108,#A9,104)
1310 CALL KEY(2,K,T):: IF T<
>0 THEN CALL MOTION(#1,0,0):
: GOTO 1370
1320 CALL PATTERN(#A9,104)
1330 IF RND<HAJK THEN CALL P
ATTERN(#11,124):: CALL SOUND
(-100,-6,7):: CALL SOUND
(-10,-3,5)
1340 IF RND<.5 THEN CALL MOT
ION(#INT(2+RND*9)+1,0,INT(1+
RND*GRAD)+1)ELSE IF RND>
.5 THEN CALL MOTION(#INT(2+R
ND*9)+1,0,-INT(1+RND*GRAD)+1
)
1350 CALL COINC(ALL,K):: IF
K<>0 THEN 1500
1360 GOTO 1280
1370 CALL POSITION(#1,B,A)::
 CALL SPRITE(#28,120,16,INT(
22*8),A)
1380 U=U-1 :: IF U<1 THEN 15
00 :: B4=INT(21*8)
1390 B4=B4-08 :: IF B4<1 THE
N 1420 :: CALL LOCATE(#28,B4
,A)
1400 CALL COINC(ALL,R):: IF
R<>0 THEN 1440
1410 GOTO 1390
1420 CALL DELSPRITE(#28)
1430 GOTO 1280
1440 PUNKTE=PUNKTE+INT(5+RND
*100)+1
1450 FOR I=2 TO 09 :: CALL C
OINC(#28,#I,20,TR):: IF TR=-
1 THEN 1480
1460 NEXT I
1470 GOTO 1490
1480 W=W+1 :: IF W=08 THEN 1
510
1490 CALL DELSPRITE(#I,#28):
: GOTO 1280
1500 CALL DELSPRITE(ALL):: C
ALL CHARSET :: GOTO 120
1510 CALL CLEAR :: CALL DELS
PRITE(ALL)
1520 CALL CHAR(98,"0000FFFF0
000FFFF")
1530 RANDOMIZE
1540 CALL CHAR(97,"FFFFFFFFF
FFFFFFF")
1550 CALL SCREEN(5):: CALL C
OLOR(1,5,5,9,7,5,14,7,5)
1560 PRINT "          a     aa
     a               a  aa
 aa  a               a
           a              a
          a     "
1570 PRINT "
                           aa
               a         a
aaaaa          a  "
1580 PRINT "  a          abbbb
a         a     a aaa    abbbb
a    aaa a     a          a
bbbba          a  '
1590 PRINT "  aaaaaaaaaaahhhh
aaaaaaaaaa



     "
1600 PRINT "  aaaaaaaaaaaaaaa
aaaaaaaaaa     a          a
 a        a     a aaa
           aaa a     a
               a
1610 PRINT "
                          "
1620 PRINT "              a
 a                        a
 a          "
1630 CALL CHAR(ASC("="),"000
3000000000000")
1640 CALL HCHAR(24,1,97,64):
: CALL VCHAR(1,32,97,48)
1650 CALL MAGNIFY(3):: CALL
CHAR(36,"FFFFFFFFFFFFFFFFFFFF
FFFFFFFFFFFFF0000000000
00000000000000000000300000000
00")
1660 CALL CHAR(40,"01030301
03070F1B1B030303060C0C0C0C0C
0A0C0E0F0DBCCC0C0C050303
05B")
1670 CALL SPRITE(#1,40,8,12*
8,16*8)
1680 E=INT(2+RND*26)+1 :: F1
=INT(2+RND*18)+1
1690 CALL GCHAR(F1+1,E,S)::
IF S<>32 THEN 1680 :: CALL G
CHAR(F1+1,E+2,P):: IF P<
>32 THEN 1680 :: CALL GCHAR(
F1-1,E-2,P1):: IF P1<>32 THE
N 1680
1700 CALL GCHAR(F1,E+1,P3)::
 IF P3<>32 THEN 1680 :: CALL
 GCHAR(F1,E-1,P4):: IF P
4<>32 THEN 1680
1710 CALL HCHAR(F1,E,ASC("="
),3):: CALL HCHAR(F1+1,E,ASC
("="),3):: CALL HCHAR(F1
-1,E,ASC("="),3)
1720 CALL CHAR(ASC("="),"00"
)
1730 CALL SPRITE(#2,36,7,12*
8,10*8,#3,36,7,14*8,10*8,#4,
36,7,16*8,10*8)
1740 CALL SPRITE(#5,36,7,18*
8,10*8)
1750 CALL CHAR(52,"01030301
0303030303030303030303080C0C
080C0C0C0C0C0C0C0C0C0C0C
0C0C0")
1760 ZE=40
1770 CALL CHAR(44,"01030301
03070F1B3030306060C0C11B0C0C
0A0C0F0F0B3DAC0C0A030303
0700000")
1780 CALL SPRITE(#7,36,7,10*
8,10*8,#8,36,7,8*8,10*E,#9,3
6,7,6*8,10*8)
1790 XDIR=DEGRI
1800 GE=40
1810 XDIR=-XDIR
1820 CALL MOTION(#2,0,XDIR,#
3,0,XDIR,#4,0,XDIR,#5,0,XDIR
,#7,0,XDIR,#8,0,XDIR,#9,
0,XDIR)
1830 IF RND<.1 THEN 1810
1840 CALL JOYST(2,X,Y):: CAL
L MOTION(#1,-Y*2,X*2)
1850 IF X=4 THEN GE=40 ELSE
IF X=-4 THEN GE=44
1860 CALL PATTERN(#1,GE)
1870 CALL POSITION(#1,B1,A1)
:: CALL GCHAR(INT(B1/8)+1.5,
INT(A1/8)+2,SP)
1880 IF SP=ASC("=")THEN 1540
 ELSE IF SP=98 THEN 1930 ELS
E IF SP<>32 THEN CALL MO
TION(#1,0,0)
1890 IF SP<>32 THEN CALL SOU
ND(-300,-5,2)
1900 CALL COINC(ALL,TR):: IF
 TR<>0 THEN 120
1910 CALL PATTERN(#1,52)
1920 GOTO 1830
1930 GOTO 2000
1940 CALL SOUND(-100,400,0):
: CALL HCHAR(1,16,140)
1950 CALL MOTION(#1,0,0)
1960 FOR I=2 TO 9 :: CALL CO
LOR(#I,16):: NEXT I
1970 TRE=1
1980 CALL CHAR(140,"1818FFFF
18181818"):: CALL COLOR(9,16
,5,14,16,5)
1990 GOTO 1830
2000 CALL CLEAR :: CALL DELS
PRITE(ALL)
2010 CALL CHAR(99,"00000001
F3F7FFF",100,"00000000F8FCFE
FF")
2020 PRINT : : : : :
2030 CALL SCREEN(5):: CALL C
OLOR(9,12,5,10,12,5)
2040 PRINT "aaaaaa
      aaaaaaa     a
      a     aa      a
            a     aa     a cd cd
 cd cd cd a      aaa a aaaaaaa
aaaaaaaaaaa     aa"
2050 PRINT " a
            a  a
            a  a
               a  a
```

```
2060 PRINT " a
          a  a        a
  a       a  a        a
     a       a  a          aaa
    aaa      a "
2070 PRINT " a
          a  a
          a  a
aaaa            a  a
abba            a "
2080 PRINT " a          abba
          a  a          abba
          a  a
             a  a
             a "
2090 PRINT " a
          a  a
          a "
2100 ORW=0
2110 CALL HCHAR(24,4,97,26)
2120 CALL SPRITE(#1,6E,8,160
,15*8,#2,40,2,50,40)
2130 ZW=40
2140 CALL POSITION(#1,B,A,#2
,B1,A1)
2150 IF INT(91) INT(A-2)THEN
 D=0T-1 :: F0=43 ELSE IF INT
(A1)>INT(A+2)THEN D=-0T-
1 :: F0=44 ELSE D=0
2160 IF B1 INT(B-2)THEN D1=0
T ELSE IF B1>INT(B-2)THEN D1
=-0T ELSE D1=0
2170 CALL PATTERN(#1,52,#2,F
0)
2180 CALL JOYST(2,,T):: CAL
L MOTION(#1,-Y*2,X*2,#2,D1,D
)
2190 CALL POSITION(#1,B,A)::
 CALL GCHAR(INT(B/8)+1,INT(A
/8)+1,DW):: IF DW=58 AND
 ORW=1 THEN 2320 ELSE IF DW=
98 THEN 1510 ELSE IF DW<>32
THEN CALL MOTION(#1,0,0)
:: CALL SOUND(200,-5,0)
2200 CALL KEY(2,K,T):: IF T<
>0 THEN 2270
2210 IF X=4 THEN ZW=40 ELSE
IF X=-4 THEN ZW=44
2220 CALL PATTERN(#1,ZW):: C
ALL PATTERN(#2,52)
2230 CAL. COINC(#1,#2,12,HG)
:: IF HG>0 THEN 120
2240 CALL MOTION(#2,DL,D)
2250 GOTO 2140
2260 GOTO 2260
2270 CALL POSITION(#1,B,A)::
 IF TRE=0 THEN 2200 ELSE IF
A<10*8 AND B<16*8 AND TR
E=1 THEN CALL SPRITE(#4,44,2
,2*8,210,0,0)ELSE TRE=0 :: G
OTO 120
2280 CALL HCHAR(5,27,97):: C
ALL HCHAR(5,6,32)
2290 CALL SPRITE(#2,44,10,2*
8,3*8
2300 ORW=1
2310 GOTO 2140
2320 CALL CHARSET :: CALL CL
EAR :: CALL DELSPRITE(ALL)::
 DISPLAY AT(5,8):"GLUECK
WUNSCH
2330 DISPLAY AT(8,5):"SIE HA
BEN IHRE FRAU GERETTET"
2340 ASD=PUNKTE*PUNKTE*U
2350 DISPLAY AT(11,06):'UND
HABEN DABEI ";STR$(INT(ASD))
;"          E
                PUNKTE GESAMMEL
T'
2360 FOR I=110 TO 200 :: CAL
L SOUND(-500,I,1):: NEXT I
2370 FOR I=1 TO 10 :: CALL S
OUND(-100,210+RND*200,3):: N
EXT I
2380 DISPLAY AT(20,1):"PRESS
 REDE OR BACK TO START!"
2390 CALL KEY(0,K,T):: IF T<
>0 THEN 110
2400 GOTO 2380
```

3D-ESCAPE

für den Spectrum 48K

Dieses Spiel ist in 3D-Version geschrieben und ermöglicht eine gute räumliche Wirkung: Sie befinden sich im obligatorischen Irrgarten und versuchen, den Ausgang zu finden. Die reizvolle Variante besteht darin, daß über die Schwierigkeiten der Orientierung hinaus auch noch ein nur begrenzt verfügbarer Luftvorrat vorhanden ist. Fünf Teile besitzt das Programm: Vorprogramm, Hauptprogramm und MC-Routinen.

Listing 4 kann nach Belieben in das Vor- oder Hauptprogramm eingefügt werden, um zu testen, ob das Labyrinth fehlerfrei eingetippt wurde. Es darf zwischen zwei Punkten im Labyrinth immer nur einen Weg geben. Außerdem muß man von jeder Stelle im Irrgarten an eine andere gelangen können.
Der Ausdruck gibt natürlich auch noch einen Überblick über die Größe des Labyrinths!

Das Maschinenprogramm:
Nachdem Listing 1 und 2 eingetippt und auf Kassette gespeichert wurde, sollte Listing 5 eingegeben werden. Dann wird mit RUN gestartet und die Zahlen von Listing 3 eingetippt. Die linke Zahl in jeder Zeile ist die Anfangsadresse der nachfolgenden 5 Bytes und dient bei fehlerhaften Eingaben leichteren Kontrollen.
Die letzten 22 Werte erscheinen immer zur Kontrolle auf dem Bildschirm. Wenn das Maschinenprogramm fehlerhaft eingetippt wurde, wird dies mitgeteilt, ansonsten zur Sicherheit auf Kassette geschrieben.

Variablenliste:

1. Zeichenketten
A$ = Tastencode
MS(2016) = Labyrinthdaten
N$ = Datas für Überschrift
WS = Hilfsstring
X$ = Tonhöhe der Noten des Trauermarsches von Chopin
Y$ = Länge der Noten des Trauermarsches
H$,J$,K$ = Hilfsstrings für das 3D Bild

2. Zahlenarrays
M(9) = Einsprungadressen in die einzelnen M-Code Routinen
N(11) = Werden als Hilfsflags benutzt

3. Variablen
i,j = für For-Next Schleifen
nn,wl = Hilfsvariablen
ex = Position des Ausgangs
x = Position des Spielers
x1 = Gehrichtung
ox = Luftvorrat
ex1,ex2 = 2 dimensionale Position des eindimensionalen Labyrinthes zur Richtungsbestimmung
oy = gefundene Luftbehälter x 2
ti = benötigte Zeit
d = doppelte Gehrichtung
as = kontrolliert das Joystickinterface
xr,xs = siehe ex1,ex2
fl = Hilfsflag
mt,m1,mb,mr = Die Daten in diesen Variablen zentrieren die Spielerfigur in die Mitte des Kartenausdruckes
mp = Adresse der linken oberen Ecke der Karte im Speicher
mp1,mp2 = Die Größe der Karte beim Ausdruck

CHOPPER-MISSION für den VC-20 mit /16-K Erw.

Sie durchbrechen zwar nicht nach lediglich 9,5 Sekunden Mach 1, dennoch kommt ein wenig 'Airwolfe'-Gefühl auf, wenn Sie mit Ihrem Helicopter über acht verschiedene Landschaften hinweg fliegen und ständig greuslichen Gefahren ausgesetzt sind...

a) Gebirge (Sie werden von Space Invaders angegriffen, die ganz und gar nicht mit sich spaßen lassen)

b) Mine (Minen-Monster werden Ihnen arg zu schaffen machen)

c) Stadt (der bösartige Magistrat setzt modernste Stufen-Raketen gegen Sie ein)

d) Kanalsystem (wenn Sie die elektrisch geladenen Wände berühren, haben wir einen Spielefreak weniger unter uns)

e) Inseln (auch der Flug über das nasse Element birgt Unheilvolles: Wasserbomben u. -Minen, Flugzeuge und Meteoriten könnten Ihnen den Garaus machen)

f) Höhle (vor den Feuerbällen, die Ihnen hier ans Leder wollen, können Sie sich lediglich in Felsspalten retten)

g) Spinnenmonster (setzen Ihnen mit Gifttropfen zu)

Gegen die möglichen Folgen dieses Horrortrips können lediglich Laser, Superbombe und Schutzschilder helfen, die Ihnen allerdings nur bei geschicktem Einsatz brauchbare Dienste leisten werden. CHOPPER-MISSION ist was für die Fans eines guten Action Games, denen es vor allem auf Reaktion und Vielseitigkeit ankommt.

RGH-BASIC für den Spectrum 48K

Das Spectrum-Basic wird von seinen Anwendern heiß und innig geliebt. Dennoch gibt es Bereiche, in denen zusätzliche Befehle, zumal für den Einsteiger, nützlich sind.

Diese Überlegungen führten einen deutschen Programmierer zu dem Entschluß hin, ein einerseits erweitertes, andererseits einfacheres Basic für den ZX Spectrum zu erstellen.

Die von Wicosoft in Kürze auf den Markt kommende Kassette mit der Bezeichnung RGH-Basic trägt diesen Forderungen Rechnung.

Das RGH-Basic ist vor allem für Spiele, die nur Integers benötigen, geeignet. Dazu tragen Spezialbefehle bei, die die Programmierung wesentlich vereinfachen. Nicht geeignet ist dieses Basic für Anwendungen, bei denen es auf Fließkomma-Darstellungen ankommt. RGH ist ein äusserst schnelles Basic, um einiges schneller als das Original ZX- oder Beta-Basic. Die hohe Geschwindigkeit wird vor allem durch die Beschränkung auf Integers und durch die Verwendung von Spezialbefehlen erreicht. Zusätzliche Befehle wie DRAWTO,CIRCLE,COLOUR,a,GRAPH, PAINT,PLOT,POS,ROLL, SCROLL,SOUND,NOISE, MUSIC,INKEY,POINT und viele mehr stehen für die Vielseitigkeit von RGH-Basic. Wie gesagt, die Anschaffung rentiert sich, wenn Sie selbst umfangreiche Spiele-Software erstellen und in Maschinensprache nicht firm genug sind.

Besonders vereinfacht wird durch das RGH-Basic die Programmierung von Grafik und Soundelementen, wie sie bei Arcade-Spielen unerläßlich sind. Auch ein Soundboard, in letzter Zeit bei der Spectrum-Anwendern besonders beliebt, kann angeschlossen und mittels RGH-Basic leicht programmiert werden.

Das Programm wird komplett mit einem wirklich umfangreichen deutschsprachigem Manual geliefert. Über den Preis war bei Redaktionsschluß noch nichts zu erfahren, dieser dürfte sich jedoch unterhalb der 100-DM-Grenze bewegen. Das Programm wird ab Ende Juli im Handel erhältlich sein.

HÖHLENJAGD für den TI-99/4A

Wollen Sie wissen, was Wumpus ist? Seien Sie lieber nicht neugierig, denn Sie lassen sich auf einen wahren Horrortrip ein. Wumpus auf verschlungenen Pfaden kreuz und quer durch ein schauriges Labyrinth ausfindig zu machen. Wumpus (wir machen das Abenteuer nicht weniger schrecklich, wenn wir das Geheimnis teilweise lüften) ist eine Stadt des Grauens aus lauter Höhlen tief unter der Erde. Selbst wenn Sie die Stadtgrenze noch nicht erreicht haben, weisen Ihnen Blutspuren unmißverständlich den Weg.

Wumpus ist jedoch nicht nur eine Stadt, sondern auch ein Monster, das es aufzuspüren gilt. Wumpus hockt in seiner Höhle, doch da sich alle ähneln, so bedarf es schon einer gehörigen Portion Spürsinn, die richtige zu treffen. Wumpus kann durch Pfeile erlegt werden, die auf Verdacht hin abzuschießen sind. Schießt man jedoch auf eine Höhle, die unbesetzt ist, so ist das Spiel verloren.

Drei Schwierigkeitsstufen sind möglich, die Bedienungsanleitung etwas knapp gehalten, doch bringen Ihnen die farbenfrohe Grafik, detailiert gestaltete Bewegungsabläufe und ein vielschichtiges Geschehen sicher eine Menge Spaß.

MELBOURNE DRAW für den Spectrum 48K

Philip Mitchell vom englischen Software-Haus MELBOURNE HOUSE hat einen Graphik-Generator geschrieben, mit dessen Hilfe Sie elegante graphische Lösungen ohne komplizierte Tastatur-Akrobatik verwirklichen können. Das Programm hilft insbesondere dann, wenn es um komfortable Bildschirmgestaltung im Zusammenhang mit Anwendersoftware und computerunterstütztem Zeichnen geht. Für die Programmierung von Spielen dürfte es nur bedingt geeignet sein.

SUPERCODE

für den TI-99/4A

**Supercode ist ein Logikspiel, welches dem bekannten Mastermind nachempfunden ist.
Man kann es zu zweit oder auch allein spielen. In maximal acht Schritten muß ein Spieler versuchen, eine verdeckte 4-Farben-Kombination zu erraten; dabei müssen Farben und Reihenfolge der geratenen Kombination mit der versteckten Kombination übereinstimmen.
Es stehen insgesamt sechs Farben zur Verfügung, die beliebig miteinander kombiniert werden können.**

Nach dem Laden des Programmes erscheint auf dem Bildschirm die Anfrage, ob allein oder zu zweit gespielt werden soll. Tippt man 1, so versteckt der Computer unsichtbar hinter der Abdeckung seinen Code. Durch ein Akustikzeichen werden Sie aufgefordert, ihren Tip abzugeben. Sollten Sie sich einmal vertippt haben, so geben Sie die vollständige Kombination ein und betätigen danach die = Taste. Die Kombination ist nun gelöscht, und Sie können eine neue eingeben.
Haben Sie alles richtig eingetippt, so drücken Sie die Space-Taste.
Nun gibt der Computer Ihnen Hilfen. Er setzt kleine weiße und schwarze Steine.
Setzt der Computer einen schwarzen Stecker, so bedeutet dies, daß ein Stein, der von Ihnen gesetzten Kombination, in Farbe und Position mit einem verstecktem Stein übereinstimmt. Setzt der Computer einen weißen Stein, so ist nur die Farbe eines Steckers auch in der versteckten Kombination enthalten, jedoch ist die Position dieses Steckers noch falsch. Setzt er keinen Stein, so ist auch keine Farbe richtig.
Das Spiel ist zu Ende, wenn Sie die Kombination erraten haben oder dies in acht Schritten nicht geschafft haben. Danach fragt der Computer Sie, ob Sie noch einmal spielen wollen.
Drücken Sie bei der Spielwahl 2, so werden Sie gebeten, die Anzahl der Runden, die Sie spielen wollen, festzulegen (bis zu 1000). Danach werden die Vornamen der Teilnehmer eingegeben. Spieler 1 übernimmt die Funktion des Computers. Er versteckt die Kombinationen, die er durch Space verdecken und durch = wieder löschen kann. Danach muß Spieler 2 die Kombinationen eingeben. Nach einem Akustikzeichen übernimmt Spieler 2 die Kommentarfunktion des Computers, wobei der Computer immer verdeckt mitspielt. Er kontrolliert die Eingaben des Spielers 1. Fehler sind nicht möglich, denn der Computer macht die Spieler sofort darauf aufmerksam. Nach einem Spieldurchgang werden die Aufgaben getauscht. Zuvor werden noch die Aufgaben der Spieler, die Anzahl der gespielten Runden und der Spielstand angezeigt.

Die Punktezählung:
Errät der Spieler die Kombination nicht, so erhält er auch keine Punkte. Errät er sie beim achten Schritt, so erhält er zwei Punkte. Für jeden Schritt, den er weniger braucht, erhält er zwei Punkte mehr. Irrt sich der andere Spieler unabsichtlich oder absichtlich(!), so erhält der ratende Spieler für jeden Irrtum nochmals vier Punkte.
Das Spiel ist dann zu Ende, wenn die gewählte Rundenzahl gespielt worden ist. Danach erfolgt wieder die Anfrage, ob noch einmal gespielt werden soll.
Aus technischen Gründen ist der Programm-Text in englischer Sprache (weil kürzer!) geschrieben.
Das Programm sollte mit niedergedrückter Alphalock-Taste programmiert und gespielt werden.

Programmablauf:
50-1650: Zeichencodedefinition
1700-2500: Farbbelegung für den Spielanfang
2550-4650: Anfragen für den späteren Spielverlauf
4700-9200: Aufbau des Spielbrettes
9250-30000: Hauptprogramm
24750: Anfrage, ob ein neues Spiel gewünscht wird

Variablenliste:
K: Rückmeldevariable in Key-Unterprogramm
S: Statusvariable in Key-Unterprogramm
A: Anzahl der Mitspieler
B: Anzahl der schwarzen Steine, die der Computer berechnet
W: Anzahl der weißen Steine, die der Computer berechnet
D: Punktestand
E: Punktestand
F: Rundenzahl
G: Fester Wert (wird für den Ausdruck der Rundenzahl benötigt)
H=: String zur Identifikation des Spielers, der die Steine versteckt
I, J: Zählvariable in FOR-NEXT-Schleifen
L=: String für den Spieler, der die Farben rät
M: Schrittanzahl
N,O,P,Q: Variable zur Berechnung von schwarzen und weißen Steinen
CL(I), C(I): Computerberechnete Farbwerte
PL1=: Name des Spielers 1
PL2=: Name des Spielers 2
SO: Zeile, wo auf Knopfdruck ein Stein erscheinen soll
SX: Spalte, wo auf Knopfdruck ein Stein erscheinen soll
SP: Momentane Zeilennummer
S1(I),ST(I): Farbcode, der vom Spieler gedr. Tasten
WB: Anzahl der schwarzen Steine, die PL1$ angibt
WW: Anzahl der weißen Steine, die PL1$angibt

```
50 REM  SUPERCODE
        FUER TI 99/4A
100 REM
150 REM   MATTHIAS NOLTE
          KLAUSBERGWEG 11
          3206 LAMSPRING
E
200 RANDOMIZE
250 CALL SCREEN(2)
300 CALL CHAR(59,"0C10202020
20100C")
350 CALL CHAR(60,"3008040404
040830")
400 CALL CHAR(61,"0044442810
101010")
450 CALL CHAR(62,"0102040810
204080")
500 CALL CHAR(91,"3C74FFFFFF
3C18")
550 CALL CHAR(92,"003C7E7E3C
1818")
600 CALL CHAR(96,"3C74FFFFFF
3C18")
650 CALL CHAR(104,"3C74FFFFF
F3C18")
700 CALL CHAR(112,"3C74FFFFF
F3C18")
750 CALL CHAR(113,"00003E42B
F423C")
800 CALL CHAR(114,"000000000
0003C3C")
850 CALL CHAR(115,"003C7E7E3
C1818")
900 CALL CHAR(120,"3C74FFFFF
F3C18")
950 CALL CHAR(128,"3C74FFFFF
F3C18")
1000 CALL CHAR(136,"80C0E0F0
F8FCFEFF")
1050 CALL CHAR(137,"FF7F3F1F
0F070301")
1100 CALL CHAR(138,"0103070F
1F3F7FFF")
1150 CALL CHAR(139,"F0F0F0F0
F8FCFEFF")
1200 CALL CHAR(140,"C0C0E0E0
E0E0E0F")
1250 CALL CHAR(141,"80808080
80C0C0C0")
1300 CALL CHAR(144,"FF")
1350 CALL CHAR(145,"80402010
08040201")
1400 CALL CHAR(146,"FF808080
80808080")
1450 CALL CHAR(147,"8040201F
08040201")
1500 CALL CHAR(148,"00000000
0F")
1550 CALL CHAR(149,"9050301F
")
1600 CALL CHAR(150,"40402020
20202020")
1650 CALL CHAR(151,"80808080
80404040")
1700 CALL CLEAR
1750 CALL COLOR(1,11,1)
1800 CALL COLOR(2,16,2)
1850 CALL COLOR(3,7,1)
1900 CALL COLOR(4,7,1)
1950 CALL COLOR(5,7,1)
2000 CALL COLOR(6,12,1)
2050 CALL COLOR(7,11,1)
2100 CALL COLOR(8,11,1)
2150 CALL COLOR(9,2,2)
2200 CALL COLOR(10,2,2)
2250 CALL COLOR(11,2,2)
2300 CALL COLOR(12,2,2)
2350 CALL COLOR(13,2,2)
2400 CALL COLOR(14,2,2)
2450 CALL COLOR(15,2,2)
2500 CALL COLOR(16,2,2)
2550 CALL SCREEN(2)
2600 PRINT "     SUPERCODE"
2650 PRINT
2700 PRINT "GAME FOR ONE(1)
OR"
2750 PRINT "TWO(2) PLAYERS ?
2800 PRINT
2850 PRINT "PLEASE PRESS 1 O
R 2!"
2900 PRINT
2950 CALL KEY(0,K,S)
3000 IF S=0 THEN 2950
3050 IF K<>49 THEN 3350
3100 CALL HCHAR(24,3,K)
3150 A=1
3200 FOR I=1 TO 200
3250 NEXT I
3300 GOTO 4700
3350 IF K<>50 THEN 2950
3400 D=0
3450 E=0
3500 CALL HCHAR(24,3,K)
3550 A=2
3600 FOR I=1 TO 200
3650 NEXT I
3700 CALL CLEAR
3750 PRINT "NUMBER OF ROUNDS
 YOU"
3800 INPUT "WANT TO PLAY
          (UP TO 1000): ":
F
3850 F=INT(F)
3900 IF F<=0 THEN 3700
3950 IF F>1000 THEN 3700
4000 F=F*2
4050 G=F/2+1
4100 PRINT
4150 PRINT "GIVE ME YOUR FIR
ST"
4200 PRINT "NAMES PLEASE!"
4250 PRINT
4300 INPUT "PLAYER 1: ":PL1$
4350 IF PL1$="" THEN 4300
4400 IF LEN(PL1$)<4 THEN 430
0
4450 INPUT "PLAYER 2: ":PL2$
4500 IF PL2$="" THEN 4450
4550 IF LEN(PL2$)<4 THEN 445
0
4600 H$=PL1$
4650 L$=PL2$
4700 CALL CLEAR
4750 CALL COLOR(2,2,2)
4800 CALL HCHAR(1,12,80)
4850 CALL HCHAR(1,13,76)
4900 CALL HCHAR(1,14,69)
4950 CALL HCHAR(1,15,65)
5000 CALL HCHAR(1,16,83)
5050 CALL HCHAR(1,17,69)
5100 CALL HCHAR(1,19,87)
5150 CALL HCHAR(1,20,65)
5200 CALL HCHAR(1,21,73)
5250 CALL HCHAR(1,22,84)
5300 IF A=2 THEN 6200
5350 CALL HCHAR(2,9,73)
5400 CALL HCHAR(2,10,39)
5450 CALL HCHAR(2,11,77)
5500 CALL HCHAR(2,13,67)
5550 CALL HCHAR(2,14,72)
5600 CALL HCHAR(2,15,79)
5650 CALL HCHAR(2,16,79)
5700 CALL HCHAR(2,17,83)
5750 CALL HCHAR(2,18,73)
5800 CALL HCHAR(2,19,78)
5850 CALL HCHAR(2,20,71)
5900 CALL HCHAR(2,22,67)
5950 CALL HCHAR(2,23,79)
6000 CALL HCHAR(2,24,76)
6050 CALL HCHAR(2,25,79)
6100 CALL HCHAR(2,26,82)
6150 CALL HCHAR(2,27,83)
6200 CALL HCHAR(23,22,146)
6250 CALL HCHAR(23,23,144,9)
6300 FOR I=7 TO 21
6350 CALL HCHAR(I,I+2,40,7)
6400 NEXT I
6450 CALL HCHAR(22,22,40,9)
6500 FOR I=1 TO 18
6550 CALL HCHAR(4+I,13+I,136
)
6600 NEXT I
6650 FOR I=1 TO 18
6700 CALL HCHAR(4+I,3+I,145)
6750 NEXT I
6800 FOR I=1 TO 18
6850 CALL HCHAR(5+I,3+I,137)
6900 NEXT I
6950 IF A=2 THEN 7700
7000 CALL HCHAR(6,6,148)
7050 CALL HCHAR(6,7,149)
7100 CALL HCHAR(6,8,148,7)
7150 CALL HCHAR(5,7,150)
7200 CALL HCHAR(4,7,151)
7250 CALL HCHAR(4,6,138)
7300 CALL HCHAR(5,6,145)
7350 CALL HCHAR(5,5,40)
7400 CALL HCHAR(6,15,139)
7450 CALL HCHAR(5,15,140)
7500 CALL HCHAR(4,15,141)
7550 CALL HCHAR(4,8,40,7)
7600 CALL HCHAR(5,8,40,7)
7650 GOTO 7950
7700 CALL HCHAR(5,5,40,9)
7750 CALL HCHAR(6,8,40,7)
7800 FOR I=1 TO 7 STEP 2
7850 CALL HCHAR(5,6+I,113)
7900 NEXT I
7950 FOR I=8 TO 14 STEP 2
8000 FOR J=1 TO 16 STEP 2
8050 CALL HCHAR(J+6,I+J,113)
8100 NEXT J
8150 NEXT I
8200 FOR I=1 TO 16
8250 CALL HCHAR(I+5,I+5,114,
2)
8300 NEXT I
8350 CALL HCHAR(1,1,32,30)
8400 CALL HCHAR(2,1,32,30)
8450 CALL COLOR(1,2,1)
8500 CALL COLOR(2,7,7)
8550 CALL COLOR(3,2,1)
8600 CALL COLOR(4,2,1)
8650 CALL COLOR(5,2,1)
8700 CALL COLOR(6,2,1)
8750 CALL COLOR(7,2,1)
8800 CALL COLOR(8,16,7)
8850 CALL COLOR(9,12,7)
8900 CALL COLOR(10,9,7)
8950 CALL COLOR(11,2,7)
9000 CALL COLOR(12,13,7)
9050 CALL COLOR(13,8,7)
9100 CALL COLOR(14,7,1)
9150 CALL COLOR(15,16,7)
9200 CALL COLOR(16,16,7)
9250 CALL SCREEN(4)
9300 IF A>=2 THEN 10400
9350 FOR I=1 TO 4
9400 C(I)=INT(6*RND)+1
9450 NEXT I
9500 FOR I=1 TO 4
9550 IF C(I)>1 THEN 9700
9600 C(I)=2
9650 GOTO 10350
9700 IF C(I)>2 THEN 9850
9750 C(I)=12
9800 GOTO 10350
9850 IF C(I)>3 THEN 10000
9900 C(I)=9
9950 GOTO 10350
10000 IF C(I)>4 THEN 10150
10050 C(I)=16
10100 GOTO 10350
10150 IF C(I)>5 THEN 10300
10200 C(I)=13
10250 GOTO 10350
10300 C(I)=8
10350 NEXT I
10400 IF A<>2 THEN 11800
10450 FOR I=1 TO 4
10500 CALL HCHAR(1,2+I,ASC(S
EG$(H$,I,4)))
10550 NEXT I
10600 CALL HCHAR(1,7,58)
10650 CALL HCHAR(1,9,83)
10700 CALL HCHAR(1,10,69)
10750 CALL HCHAR(1,11,84)
10800 FOR I=1 TO 4
10850 CALL HCHAR(2,2+I,ASC(S
EG$(L$,I,4)))
10900 NEXT I
10950 CALL HCHAR(2,7,58)
11000 CALL HCHAR(2,9,71)
11050 CALL HCHAR(2,10,85)
11100 CALL HCHAR(2,11,69)
11150 CALL HCHAR(2,12,83,2)
11200 CALL HCHAR(20,3,82)
11250 CALL HCHAR(20,4,79)
11300 CALL HCHAR(20,5,85)
11350 CALL HCHAR(20,6,78)
11400 CALL HCHAR(20,7,68)
11450 IF F/2<>INT(F/2)THEN 1
1650
11500 FOR I=1 TO LEN(STR$(G-
F/2))
11550 CALL HCHAR(20,8+I,ASC(
SEG$(STR$(G-F/2),I,LEN(STR$(
G-F/2)))))
11600 NEXT I
11650 SO=7
11700 SX=5
11750 GOTO 11950
11800 M=0
11850 SO=23
11900 SX=21
11950 LET SP=SO
12000 CALL SOUND(100,292,0)
12050 FOR I=1 TO 4
12100 CALL KEY(0,K,S)
12150 IF S=0 THEN 12100
12200 ST(I)=K-48
12250 IF ST(I)<1 THEN 12100
12300 IF ST(I)>6 THEN 12100
12350 ON ST(I)GOTO 12450,126
00,12750,12900,13050,13200
12400 GOTO 12100
12450 ST(I)=2
12500 CALL HCHAR(SX,SP,112)
12550 GOTO 13300
12600 ST(I)=12
12650 CALL HCHAR(SX,SP,96)
12700 GOTO 13300
12750 ST(I)=9
12800 CALL HCHAR(SX,SP,104)
12850 GOTO 13300
12900 ST(I)=16
12950 CALL HCHAR(SX,SP,91)
13000 GOTO 13300
13050 ST(I)=13
13100 CALL HCHAR(SX,SP,120)
13150 GOTO 13300
13200 ST(I)=8
13250 CALL HCHAR(SX,SP,128)
13300 SP=SP+2
13350 IF SP=SO+8 THEN 13450
13400 NEXT I
13450 CALL KEY(0,K,S)
13500 IF S=0 THEN 13450
13550 IF K=61 THEN 13650
13600 IF K=32 THEN 13800 ELS
E 13450
13650 FOR I=0 TO 6 STEP 2
13700 CALL HCHAR(SX,SO+I,113
)
13750 NEXT I
13800 IF A<2 THEN 14300
13850 IF A=4 THEN 14250
13900 A=3
13950 IF K<>32 THEN 14250
14000 FOR I=1 TO 4
14050 LET C(I)=ST(I)
14100 NEXT I
14150 A=4
14200 GOTO 7000
14250 IF K=61 THEN 11950
14300 IF K<>61 THEN 14400
14350 GOTO 11950
14400 B=0
14450 M=M+1
14500 P=30
14550 Q=100
14600 FOR I=1 TO 4
14650 LET CL(I)=C(I)
14700 LET S1(I)=ST(I)
14750 IF ST(I)<>C(I)THEN 151
50
14800 LET CL(I)=C(I)
```

```
14850 LET S1(I)=ST(I)
14900 B=B+1
14950 LET S1(I)=P
15000 LET CL(I)=Q
15050 P=P+2
15100 Q=Q+2
15150 NEXT I
15200 W=0
15250 N=40
15300 O=70
15350 FOR I=1 TO 4
15400 FOR J=1 TO 4
15450 IF S1(I)<>CL(J)THEN 15
750
15500 W=W+1
15550 S1(I)=N
15600 CL(J)=O
15650 N=N+1
15700 O=O+1
15750 NEXT J
15800 NEXT I
15850 IF A=1 THEN 20100
15900 WB=0
15950 WW=0
16000 CALL SOUND(100,392,0)
16050 CALL KEY(0,K,S)
16100 IF S=0 THEN 16050
16150 IF K=32 THEN 19000
16200 IF K<>49 THEN 16350
16250 Z=115
16300 GOTO 16450
16350 IF K=52 THEN 16400 ELS
E 16050
16400 Z=92
16450 CALL HCHAR(SX-1,SO-3,Z
)
16500 IF Z=115 THEN 16550 EL
SE 16650
16550 WB=WB+1
16600 GOTO 16700
16650 WW=WW+1
16700 CALL KEY(0,K,S)
16750 IF S=0 THEN 16700
16800 IF K=32 THEN 19000
16850 IF K=49 THEN 16900 ELS
E 17000
16900 Z=115
16950 GOTO 17100
17000 IF K=52 THEN 17050 ELS
E 16700
17050 Z=92
17100 CALL HCHAR(SX-1,SO-2,Z
)
17150 IF Z=115 THEN 17200 EL
SE 17300
17200 WB=WB+1
17250 GOTO 17350
17300 WW=WW+1
17350 CALL KEY(0,K,S)
17400 IF S=0 THEN 17350
17450 IF K=32 THEN 19000
17500 IF K=49 THEN 17550 ELS
E 17650
17550 Z=115
17600 GOTO 17750
17650 IF K=52 THEN 17700 ELS
E 17350
17700 Z=92
17750 CALL HCHAR(SX,SO-2,Z)
17800 IF Z=115 THEN 17850 EL
SE 17950
17850 WB=WB+1
17900 GOTO 18000
17950 WW=WW+1
18000 CALL KEY(0,K,S)
18050 IF S=0 THEN 18000
18100 IF K=32 THEN 19000
18150 IF K=49 THEN 18200 ELS
E 18300
18200 Z=115
18250 GOTO 18400
18300 IF K=52 THEN 18350 ELS
E 18000
18350 Z=92
18400 CALL HCHAR(SX,SO-1,Z)
18450 IF Z=115 THEN 18500 EL
SE 18600
18500 WB=WB+1
18550 GOTO 18650
18600 WW=WW+1
18650 CALL KEY(0,K,S)
18700 IF S=0 THEN 18650
18750 IF K=32 THEN 19000 ELS
E 18800
18800 IF K=61 THEN 18850 ELS
E 18650
18850 CALL HCHAR(SX-1,SO-3,1
14,2)
18900 CALL HCHAR(SX,SO-2,114
,2)
18950 GOTO 15900
19000 IF WB<>B THEN 19150 EL
SE 19050
19050 IF WW<>W THEN 19150 EL
SE 19100
19100 IF WB=4 THEN 20700 ELS
E 27900
19150 IF L$<>PL1$ THEN 19300
19200 D=D+4
19250 GOTO 19350
19300 E=E+4
19350 CALL HCHAR(1,3,68)
19400 CALL HCHAR(1,4,79)
19450 CALL HCHAR(1,5,78)
19500 CALL HCHAR(1,6,39)
19550 CALL HCHAR(1,7,84)
19600 CALL HCHAR(1,9,67)
19650 CALL HCHAR(1,10,72)
19700 CALL HCHAR(1,11,69)
19750 CALL HCHAR(1,12,65)
19800 CALL HCHAR(1,13,84)
19850 CALL HCHAR(1,15,33)
19900 FOR I=1 TO 300
19950 NEXT I
20000 CALL HCHAR(1,1,32,32)
20050 GOTO 18850
20100 IF B>0 THEN 20200
20150 GOTO 27200
20200 IF B>1 THEN 20350
20250 CALL HCHAR(SX-1,SO-3,1
15)
20300 GOTO 27200
20350 IF B>2 THEN 20500
20400 CALL HCHAR(SX-1,SO-3,1
15,2)
20450 GOTO 27200
20500 IF B>3 THEN 20700
20550 CALL HCHAR(SX-1,SO-3,1
15,2)
20600 CALL HCHAR(SX,SO-2,115
)
20650 GOTO 27200
20700 CALL HCHAR(1,16,71)
20750 CALL HCHAR(1,17,79)
20800 CALL HCHAR(1,18,79)
20850 CALL HCHAR(1,19,68)
20900 CALL HCHAR(1,21,33)
20950 CALL HCHAR(2,12,84)
21000 CALL HCHAR(2,13,72)
21050 CALL HCHAR(2,14,65)
21100 CALL HCHAR(2,15,84)
21150 CALL HCHAR(2,16,39)
21200 CALL HCHAR(2,17,83)
21250 CALL HCHAR(2,19,82)
21300 CALL HCHAR(2,20,73)
21350 CALL HCHAR(2,21,71)
21400 CALL HCHAR(2,22,72)
21450 CALL HCHAR(2,23,84)
21500 CALL HCHAR(2,25,33)
21550 CALL HCHAR(4,6,32,10)
21600 CALL HCHAR(5,14,136)
21650 CALL HCHAR(5,15,32)
21700 CALL HCHAR(6,5,145)
21750 CALL HCHAR(6,6,40,9)
21800 CALL HCHAR(6,15,136)
21850 CALL HCHAR(5,6,40,2)
21900 PO=7
21950 FOR I=1 TO 4
22000 IF C(I)<>2 THEN 22150
22050 CALL HCHAR(5,PO,112)
22100 GOTO 22800
22150 IF C(I)<>12 THEN 22300
22200 CALL HCHAR(5,PO,96)
22250 GOTO 22800
22300 IF C(I)<>9 THEN 22450
22350 CALL HCHAR(5,PO,104)
22400 GOTO 22800
22450 IF C(I)<>16 THEN 22600
22500 CALL HCHAR(5,PO,91)
22550 GOTO 22800
22600 IF C(I)<>13 THEN 22750
22650 CALL HCHAR(5,PO,120)
22700 GOTO 22800
22750 CALL HCHAR(5,PO,128)
22800 PO=PO+2
22850 NEXT I
22900 FOR I=1 TO 400
22950 NEXT I
23000 CALL HCHAR(1,1,32,32)
23050 CALL HCHAR(2,1,32,32)
23100 IF A=1 THEN 24750
23150 A=2
23200 FOR I=1 TO 4
23250 CALL HCHAR(23,4+I,ASC(
SEG$(PL1$,I,4)))
23300 CALL HCHAR(23,15+I,ASC
(SEG$(PL2$,I,4)))
23350 NEXT I
23400 IF L$<>PL1$ THEN 23500
23450 D=D+18-M*2
23500 FOR I=1 TO LEN(STR$(D)
)
23550 CALL HCHAR(24,2+I,ASC(
SEG$(STR$(D),I,LEN(STR$(D)))
))
23600 NEXT I
23650 IF L$<>PL2$ THEN 23750
23700 E=E+18-M*2
23750 FOR I=1 TO LEN(STR$(E)
)
23800 CALL HCHAR(24,13+I,ASC
(SEG$(STR$(E),I,LEN(STR$(E))
)))
23850 NEXT I
23900 CALL HCHAR(24,9,80)
23950 CALL HCHAR(24,10,84)
24000 CALL HCHAR(24,11,83)
24050 CALL HCHAR(24,20,80)
24100 CALL HCHAR(24,21,84)
24150 CALL HCHAR(24,22,83)
24200 F=F-1
24250 IF F=0 THEN 24750
24300 IF H$<>PL1$ THEN 24550
24350 H$=PL2$
24400 L$=PL1$
24450 A=2
24500 GOTO 10450
24550 H$=PL1$
24600 L$=PL2$
24650 A=2
24700 GOTO 10450
24750 CALL HCHAR(1,3,68)
24800 CALL HCHAR(1,4,79)
24850 CALL HCHAR(1,6,61)
24900 CALL HCHAR(1,7,79)
24950 CALL HCHAR(1,8,85)
25000 CALL HCHAR(1,10,87)
25050 CALL HCHAR(1,11,65)
25100 CALL HCHAR(1,12,78)
25150 CALL HCHAR(1,13,84)
25200 CALL HCHAR(1,15,84)
25250 CALL HCHAR(1,16,79)
25300 CALL HCHAR(1,18,80)
25350 CALL HCHAR(1,19,76)
25400 CALL HCHAR(1,20,65)
25450 CALL HCHAR(1,21,61)
25500 CALL HCHAR(2,3,83)
25550 CALL HCHAR(2,4,85)
25600 CALL HCHAR(2,5,80)
25650 CALL HCHAR(2,6,69)
25700 CALL HCHAR(2,7,82)
25750 CALL HCHAR(2,8,67)
25800 CALL HCHAR(2,9,79)
25850 CALL HCHAR(2,10,68)
25900 CALL HCHAR(2,11,69)
25950 CALL HCHAR(2,13,65)
26000 CALL HCHAR(2,14,71)
26050 CALL HCHAR(2,15,65)
26100 CALL HCHAR(2,16,73)
26150 CALL HCHAR(2,17,78)
26200 CALL HCHAR(2,18,59)
26250 CALL HCHAR(2,19,61)
26300 CALL HCHAR(2,20,62)
26350 CALL HCHAR(2,21,78)
26400 CALL HCHAR(2,22,60)
26450 CALL HCHAR(2,24,63)
26500 CALL KEY(0,K,S)
26550 IF S=0 THEN 26500
26600 IF K=89 THEN 26650 ELS
E 26850
26650 CALL HCHAR(2,26,61)
26700 FOR I=1 TO 100
26750 NEXT I
26800 GOTO 1700
26850 IF K=78 THEN 26900 ELS
E 26500
26900 CALL HCHAR(2,26,K)
26950 FOR I=1 TO 100
27000 NEXT I
27050 CALL HCHAR(1,1,32,30)
27100 CALL HCHAR(2,1,32,30)
27150 IF K<>74 THEN 29950
27200 IF W>0 THEN 27300
27250 GOTO 27900
27300 IF W>1 THEN 27450
27350 CALL HCHAR(SX,SO-1,92)
27400 GOTO 27900
27450 IF W>2 THEN 27600
27500 CALL HCHAR(SX,SO-2,92,
2)
27550 GOTO 27900
27600 IF W>3 THEN 27800
27650 CALL HCHAR(SX-1,SO-2,9
2)
27700 CALL HCHAR(SX,SO-2,92,
2)
27750 GOTO 27900
27800 CALL HCHAR(SX-1,SO-3,9
2,2)
27850 CALL HCHAR(SX,SO-2,92,
2)
27900 IF M<>8 THEN 29600
27950 CALL HCHAR(1,3,84)
28000 CALL HCHAR(1,4,72)
28050 CALL HCHAR(1,5,69)
28100 CALL HCHAR(1,7,82)
28150 CALL HCHAR(1,8,73)
28200 CALL HCHAR(1,9,71)
28250 CALL HCHAR(1,10,72)
28300 CALL HCHAR(1,11,84)
28350 CALL HCHAR(1,13,67)
28400 CALL HCHAR(1,14,79)
28450 CALL HCHAR(1,15,77)
28500 CALL HCHAR(1,16,66)
28550 CALL HCHAR(1,17,73)
28600 CALL HCHAR(1,18,78)
28650 CALL HCHAR(1,19,65)
28700 CALL HCHAR(1,20,84)
28750 CALL HCHAR(1,21,73)
28800 CALL HCHAR(1,22,79)
28850 CALL HCHAR(1,23,78)
28900 CALL HCHAR(1,25,87)
28950 CALL HCHAR(1,26,79)
29000 CALL HCHAR(1,27,85)
29050 CALL HCHAR(1,28,76)
29100 CALL HCHAR(1,29,68)
29150 CALL HCHAR(2,3,72)
29200 CALL HCHAR(2,4,65)
29250 CALL HCHAR(2,5,86)
29300 CALL HCHAR(2,6,69)
29350 CALL HCHAR(2,8,66)
29400 CALL HCHAR(2,9,69,2)
29450 CALL HCHAR(2,11,78)
29500 CALL HCHAR(2,13,58)
29550 GOTO 21550
29600 SX=SX-2
29650 SO=SO-2
29700 FOR I=1 TO 4
29750 LET CL(I)=C(I)
29800 LET S1(I)=ST(I)
29850 NEXT I
29900 GOTO 11950
29950 CALL CLEAR
30000 END
```

Pac Man

für den TI-99/4A

Das Programm hat einen Speicherumfang von ca. 9 K. Nach der Eingabe von "RUN" wird man von einem eindrucksvollen Einleitungstitelbild überrascht, von dem man aufgefordert wird, den Action-Knopf des Joysticks (1) zu betätigen. Dabei muß man sich aber entscheiden, welche Farbe das Labyrinth im späteren Spiel haben soll. Diese richtet sich nämlich nach der augenblicklichen Farbe der Schrift im Titel. Nach dem Drücken wird man erinnert, die richtige Stellung der "Alpha-Lock"-Taste zu überprüfen. Im Anschluß daran kann man sich an einer kleinen Sound-Einlage erfreuen, diese aber auch überspringen, indem man vom Bildschirmaufbau an bis zum Ertönen eines akustischen Signals die Feuertaste gedrückt hält (Programmzeile 450). Ein neues Bild erscheint, wenn alle Punkte verspeist wurden. Das Spiel ist beendet, wenn alle Pac-Men verspeist wurden.

Variablenliste:
C(xx): Farben
T(xx): Tonleiter
PCM: Credit
BLD: Bild
SC: Score
HSC: Highscore
LVL: Level
TMP: Geschwindigkeitsfaktor (Hier sollte nichts verändert werden!)
S: in FOR-NEXT-Schleifen & Sprites
FN, FN1, FN2: in FOR-NEXT-Schleifen
P: Anzahl der gefressenen Punkte
K, S: Verwendung in "CALL KEY(...)"
R1, R2: Verwendung in "READ"
PT: Verwendung in "CALL PATTERN (..)"
Z: Verwendung in "CALL GCHAR(...)"
X, Y: Koordination im "24*32-Raster"
X1, Y1: Koordination im "192*256-Raster"
X2, Y2: mit "CALL JOYST(...)"
PB: Anzahl der Punkte im jeweiligen Bild
HG: Höchstgeschwindigkeit der Geister im j. Bild
P1, P2: Variablen im Upro "CALL PAUSE(.)"

Strings:
A$: Labyrinthspeicher
B$: Blinkmechanismus im Vorspann

```
10 !                          !      PAC-MAN          !
   !
20 !    ON TI-99/4A
30 !    ~8.9 K-BYTE
40 !  MAY/JUNE 1984  BY
50 !    HENRIK BAUR
60 !  SCHULSTRASSE 16
70 !    7913 SENDEN
80 !  TEL: 07307/5184
90 !
100 !                         ! DIMENSIONIERUNG & RAN-
! DOMIZE
   !
110 DIM C(11),T(12)
120 RANDOMIZE
130 !                         ! VORBEREITUNG DES BILD-
! SCHIRMS
   !
140 CALL DELSPRITE(ALL):: CALL CLEAR :: CALL SCREEN(2):: CAL
L MAGNIFY(3)
150 !                         ! VARIABLEN-BESETZUNG
   !
160 C(1)=13 :: C(2)=14 :: C(3)=5 :: C(4)=16 :: C(5)=9 :: C(6
)=4 :: C(7)=11 :: C(
8)=8 :: C(9)=6 :: C(10)=10 :: C(11)=11
170 T(1)=195 :: T(2)=220 :: T(3)=246 :: T(4)=261 :: T(5)=293
 :: T(6)=329 :: T(7)
=369 :: T(8)=391 :: T(9)=440 :: T(10)=493 :: T(11)=523
180 T(12)=587 :: T(0)=277
190 PCM=5 :: BLD=1 :: SC=0 :: LVL=0 :: TMP=1.3
200 !                         ! GRAFIK
! (ASCII 59,92-143)
   !
210 CALL CHAR(59,"001C3E7B6F703E1C")
220 CALL CHAR(92,"0001070F0303121F1D0917190A02000000E0F8FCFC
3E1E1CFCF8A040C08000
00")
230 CALL CHAR(96,"00000030F0C1818181808C0F0300000000000C0F0
30181818183F0C00000
00")
240 CALL CHAR(100,"1818181818181818000000FFFF0000000000000
0000000000000000000
000")
250 CALL CHAR(104,"0000001818000000000001818000000000000000
0000000000000000000
000")
260 CALL CHAR(108,"00000001070F0F1F1C0E0F0701000000000000B0E
03030FB000080E080006
000")
270 CALL CHAR(112,"0000000107 0F09191F0D0E0701000000000000080E
0F09098FBB070E080000
000")
280 CALL CHAR(116,"0000000107 0C0C1F0000010701000000000000080E
0F0F0F83870F0E080000
000")
290 CALL CHAR(120,"0000010303070707070706070F0A1A00000000E0F09
898F8C80020F8F0A0B00
000")
300 CALL CHAR(124,"000003070F09090F0F0A0D0F0D0800000000C0E0F
09090F0F050B0F0B0100
000")
310 CALL CHAR(128,"0000070F19191F1300041F0F050D0000000080C0C
0E0E0E0E060E0F050580
000")
320 CALL CHAR(132,"0000000001031E0F001E0F0500000000000000C
07820F04060E05000000
000")
330 CALL CHAR(136,"000000000F0505070C1B0F0A0000000000000000F
0A0A0E030D8F05000000
000")
340 CALL CHAR(140,"0000000003 1E040F0206070A0000000000000008
0C078F00078F0A000000
000")
350 !                         ! FARBE
   !
360 CALL COLOR(1,6,1,2,6,1,3,11,1,4,11,1,5,14,1,6,14,1,7,14,
1,8,14,1,9,1,1,10,15
,1)
370 GOTO 990 ! ZUM VORSPANN
380 !                         ! SETZEN GANG-GEISTER
   !
390 S=0 :: FOR FN=30 TO 158 STEP 32 :: S=S+1 :: CALL SPRITE(
#S+1,124,C(S),FN,RND
*255+1,0,0):: NEXT FN
400 !                         ! SETZEN ZWISCHENGEISTER
   !
410 FOR FN=14 TO 174 STEP 32 :: S=S+1 :: CALL SPRITE(#S+1,13
6,C(S),FN,RND*255+1,
0,0):: NEXT FN
420 CALL PAUSE(200)
430 !                         ! ZUM LABYRINTHAUFBAU &
! BEWEGUNG DER GEIST
ER    !
440 P=0 :: ON BLD GOSUB 780,800,820,840,860
450 CALL KEY(1,K,S):: IF K=18 THEN 520 !LIED WIRD UEBER-SPRU
NGEN
460 !                         ! SONG
   !
470 FOR FN1=1 TO 2 :: RESTORE 1390 :: FOR FN2=1 TO 25
480 READ R1,R2 :: CALL SOUND(R1*100,T(R2),0):: NEXT FN2 :: N
EXT FN1
490 RESTORE 1420 :: FOR FN1=1 TO 52
500 READ R1,R2 :: CALL SOUND(R1*100,T(R2),0):: NEXT FN1
510 !                         ! SETZEN PAC-MAN-SPRITE
   !
520 CALL SPRITE(#1,112,11,1,125,0,0):: CALL PAUSE(50)
530 GOSUB 1460
540 !                         ! SPIELSCHLEIFE :
! RAUF-RUNTER
   !
550 PT=112
560 GOSUB 680
570 IF Z<>105 AND Z<>107 THEN 590
580 IF X2<>0 AND Y2=0 THEN CALL SPRITE(#1,PT,11,X*8-10,Y*8-1
1,0,X2*TMP):: GOTO 6
10
590 CALL MOTION(#1,-Y2*TMP,0):: GOTO 550
600 !                         ! SPIELSCHLEIFE :
! RECHTS-LINKS
   !
610 IF X2=4 THEN PT=108
620 IF X2=-4 THEN PT=116
630 GOSUB 680
640 IF Z<>105 AND Z<>107 THEN 660
650 IF Y2<>0 AND X2=0 THEN CALL SPRITE(#1,PT,11,X*8-10,Y*8-1
1,-Y2*TMP,0):: GOTO
550
660 CALL MOTION(#1,0,X2*TMP):: GOTO 610
670 !                         ! UPRO ZU DEN BEIDEN
! SPIELSCHLEIFEN
   !
680 CALL POSITION(#1,X1,Y1):: CALL JOYST(1,X2,Y2):: CALL PAT
TERN(#1,PT)
690 CALL COINC(ALL,CO):: IF CO=-1 THEN 1200
700 X=INT((X1+14)/8):: Y=INT((Y1+14)/8)
710 IF X<1 OR X>24 THEN 760
720 IF Y<1 OR Y>32 THEN 760
730 CALL GCHAR(X,Y,Z)
740 IF Z<106 AND Z>103 THEN CALL HCHAR(X,Y,Z+2):: P=P+1 :: S
C=SC+5 :: GOSUB 1370
750 IF P>=PB THEN 440
760 CALL COINC(ALL,CO):: IF CO=-1 THEN 1200 ELSE RETURN
770 !                         ! BILDSCHIRMAUFBAU
   !
780 A$=" hhhhhhhhhhhhhhihhhhhhhhhhhhh 'eeeeeeeeeeeh'eeeeeeee
eeeb d           dhd
```

```
        d aeeeeeeeeeeechaeeeeeeeeeec"
790 PB=153 :: HG=8 :: GOTO 880
800 A$=" hhhhhhhihhhhhhhihhhhhhhhh 'eeeebh'eeeebh'eeeebh'e
eeeb d    dhd    dhd
    dhd    d aeeeechaeeeechaeeeechaeeeec"
810 PB=189 :: HG=11 :: GOTO 880
820 A$=" hhihhhhihhhhhhihhhhhihhhhih 'bh'eebh'eeebh'eeebh'ee
bh'b ddhd  dhd   dhd
   dhd  dhdd achaeechaeeechaeeechaeechac"
830 PB=225 :: HG=13 :: GOTO 880
840 A$=" hhihhhhihhhhihhhihhhhihhhih 'bh'ebh'ebh'bh'bh'ebh'e
bh'b ddhd dhd dhddhd
dhd dhd dhdd achaechaechachachaechaechac"
850 PB=261 :: HG=13 :: GOTO 880
860 A$=" hihhihhhihhhihhhihhhihhhih bh'bh'bh'bh'bh'bh'bh'bh
'bh' dhddhddhddhddhd
dhddhddhddhd chachachachachachachacha"
870 PB=297 :: HG=17 :: PCM=PCM+1
880 CALL DELSPRITE(#1):: FOR FN=1 TO 6 :: PRINT A$;:: NEXT F
N :: LVL=LVL+1 :: GO
SUB 1340
890 IF BLD>=5 THEN BLD=1 ELSE BLD=BLD+1
900 FOR S=2 TO 6 :: ON INT(RND*2)+1 GOTO 910,920
910 CALL PATTERN(#S,120):: CALL MOTION(#S,0,HG-RND*HG/3):: G
OTO 930
920 CALL PATTERN(#S,128):: CALL MOTION(#S,0,-HG+RND*HG/3)::
GOTO 930
930 NEXT S
940 FOR S=7 TO 12 :: ON INT(RND*2)+1 GOTO 950,960
950 CALL PATTERN(#S,140):: CALL MOTION(#S,0,LVL*2):: GOTO 97
0
960 CALL PATTERN(#S,132):: CALL MOTION(#S,0,LVL*-2):: GOTO 9
70
970 NEXT S :: RETURN
980 !                          ! VORSPANN
 !
990 B$(1)="0000001818"
1000 B$(2),B$(6)="0000183C3C18"
1010 B$(3),B$(5)="00183C7E7E3C18"
1020 B$(4)="187E7EFFFF7E7E18"
1030 PRINT " fff  ff  ff  f  f ff f  f   f  ff  ff  f fffff
 ff  f   f  ff  ff
   f  ff  fff f   fff f  ff    f  ff  ff ff  "
1040 PRINT " f   fffff    f  ffffff  f   f   f  ff  f f  ff
 ff  f   f   f  f ff
   f  ff  ff  f  "
1050 PRINT :: PRINT "'eeeeeeeeeeeeeeeeeeeeeeeeeb"
1060 FOR I=1 TO 10 :: PRINT "d                              d" :
: NEXT I :: PRINT "a
eeeeeeeeeeeeeeeeeeeeeeeeeec" :: PRINT
1070 DISPLAY AT(14,5)SIZE(20):"PRESS FIRE TO BEGIN!"
1080 DISPLAY AT(20,3)SIZE(24):"COPYRIGHT 1984 ATAC & CO"
1090 CALL SPRITE(#1,108,11,118,154,0,20,82,140,6,118,112,0,2
0,83,120,14,118,80,0
,20)
1100 CALL SPRITE(#4,116,11,134,102,0,-20,#5,132,10,134,142,0
,-20,#6,128,7,134,17
4,0,-20)
1110 CALL SPRITE(#7,92,16,38,99)
1120 FOR FN1=1 TO 11 :: CALL COLOR(9,C(FN1),1)
1130 FOR FN2=1 TO 6 :: CALL CHAR(102,B$(FN2))
1140 CALL KEY(1,K,S):: IF K=18 THEN 1160
1150 NEXT FN2 :: NEXT FN1 :: GOTO 1120
1160 CALL DELSPRITE(ALL):: GOSUB 1460 :: FOR FN=1 TO 7 :: PR
INT :: NEXT FN
1170 PRINT "       'eeeeeeeeeeb                dALPHA LOCK?
d             aeee
eeeeeeeec     "
1180 FOR FN=1 TO 12 :: PRINT :: NEXT FN :: GOTO 390
1190 !                          ! PAC-MAN TOT
 !
1200 CALL POSITION(#1,X1,Y1):: CALL SPRITE(#1,92,16,X1,Y1,0,
0)
1210 CALL SOUND(250,184,8,369,3,739,10):: CALL SOUND(125,195
,8,391,3,783,10)
1220 CALL SOUND(125,184,8,369,3,738,10):: CALL SOUND(250,146
,8,293,3,587,10)
1230 CALL SOUND(250,138,8,277,3,554,10):: CALL SOUND(1000,12
3,8,246,3,493,10)
1240 PCM=PCM-1 :: IF PCM=0 THEN 1260 ELSE GOSUB 1340 :: GOTO
 520
1250 !                          ! "GAME OVER"
 !
1260 HSC=MAX(SC,HSC):: GOSUB 1340
1270 DISPLAY AT(12,6)SIZE(19):"'eeeeeeeeeeeeeeeeb"
1280 DISPLAY AT(13,6)SIZE(19):"d                 d"
1290 DISPLAY AT(14,6)SIZE(19):"aeeeeeeeeeeeeeeeec"
1300 FOR FN=1 TO 20 :: DISPLAY AT(13,7)SIZE(17):"G A M E   O
 V E R"
1310 CALL PAUSE(70):: DISPLAY AT(13,7)SIZE(17):"
                "
1320 CALL PAUSE(70):: NEXT FN :: GOTO 100
1330 !                          ! AUSDRUCKE
 !
1340 DISPLAY AT(1,2)SIZE(6):" ";PCM
1350 DISPLAY AT(1,16)SIZE(8):HSC
1360 DISPLAY AT(1,24)SIZE(5):"L";LVL
1370 DISPLAY AT(1,8)SIZE(8):SC
1380 RETURN
1390 !                          ! LIED-DATEN
 !
1400 DATA 1,5,1,3,2,1,2,3,2,5,4,8,1,10,1,9,2,8,2,3,2,0,4,5,1
,5,1,5,3,10,1,9,2,8
1410 DATA 4,7,1,6,1,7,2,8,2,8,2,5,2,3,2,1
1420 DATA 1,10,1,10,2,10,2,11,2,12,4,12,1,11,1,10,2,9,2,10,2
,11,4,11,2,11
1430 DATA 4,10,1,9,1,8,4,7,1,6,1,7,2,8,2,4,2,0,4,5,2,5,2,8,2
,8,1,8,1,7,2,6,2,6,2
,6
1440 DATA 2,9,1,11,1,10,1,9,1,8,2,8,2,7,1,5,1,5,2,8,1,8,1,9,
1,10,1,11,2,12,2,10,
2,8
1450 DATA 3,10,1,11,2,9,4,8
1460 !                          ! UPRO TONLEITER
 !
1470 FOR FN=1 TO 12 :: CALL SOUND(-50,T(FN),0):: NEXT FN
1480 FOR FN=11 TO 1 STEP -1 :: CALL SOUND(-50,T(FN),0):: NEX
T FN :: RETURN
1490 !                          ! UPRO PAUSE
 !
1500 SUB PAUSE(P1):: FOR P2=0 TO P1 :: NEXT P2 :: SUBEND ::
END
1510 !                          ! BY HENRIK BAUR
 !
```

TERMIN-Kalender in CPU

Name der Veranstaltung	Datum	Ort
Kasseler Computertage '84	30.08. - 02.09.84	Kassel
4. Commodore-Fachausstellung	06.09. - 08.09.84	Frankfurt/M
Elektronik-Comp.-Tage Saar	07.09. - 09.09.84	Saarbrücken
12th ZX-Microfair '84	nur 08.09.84	London
Fachtagung d. Ges. f. Informatik	08.10. - 10.10.84	Berlin
Photokina '84	10.10. - 16.10.84	Köln
Hobby-Elektronik '84	17.10. - 21.10.84	Stuttgart
Computertage Hückelhoven	20.10. - 21.10.84	Hückelhoven
Hobby + Elektronik '84	15.11. - 18.11.84	Wien
Bayerischer Computertag	nur 02.12.84	München
Hessischer Computertag	nur 09.12.84	Neu-Isenburg
Micro-Computer '85	29.01. - 03.02.85	Frankfurt/M
Computer & Communication '85	15.05. - 19.05.85	Sindelfingen

Kalenderprogramm

für den TI-99/4A

Nach dem Start des Programmes erscheint das Titelbild auf dem Bildschirm. Geben Sie in numerischer Form Tag, Monat und Jahr des heutigen Datums ein. Nach der Eingabe die ENTER-Taste drücken.

Auf dem Bildschirm erscheint nun das Menue. Wählen Sie eines der 4 Teilprogramme:
1. Monatskalender
2. Feiertage in Bayern
3. Wochentag
4. Distanz

Drücken Sie die entsprechende Taste und danach die ENTER-Taste.

Erklärung der 4 Teilprogramme:
Teilprogramm 1:
Monatskalender
Geben Sie den gewünschten Monat ein (Monatsname ausschreiben z.B. Januar) und drücken Sie danach die ENTER-Taste. Geben Sie nun das gewünschte Jahr ein (4-stellig z.B. 1983). Danach die ENTER-Taste drücken.

Der Rechner schreibt nun das Kalenderblatt aus. Die Sonn- und Feiertage werden mit roten Ziffern geschrieben. Danach werden die in diesem Monat vorkommenden Feiertage aufgelistet. Mit dem Drücken der Taste 9 wird in das Menue zurückgesprungen. Mit dem Drücken der Taste 8 wird das Teilprogramm wiederholt. Auch in den anderen Teilprogrammen sind diese Tastenfunktionen programmiert.

Teilprogramm 2:
Feiertage in Bayern
Geben Sie das gewünschte Jahr ein (4-stellig z.B. 1983). Danach die ENTER-Taste drücken. Es werden die 18 wichtigsten Feiertage (gültig für Bayern) mit Datum und Wochentag aufgelistet.

Teilprogramm 3:
Wochentage
Geben Sie das gewünschte Datum ein. Das Eingabeformat beachten (z.B. 01.01.1983). Danach die ENTER-Taste drücken. Nun wird der Wochentag für das eingegebene Datum ausgeschrieben.

Teilprogramm 4:
Distanz
Geben Sie das erste Datum ein (Eingabeformat wie bei Teilprogramm 3). Danach die ENTER-Taste drücken. Geben Sie das zweite Datum ein und drücken Sie die ENTER-Taste. Der Computer gibt nun die Differenz der beiden Daten in Tagen aus.

```
100 !   KALENDERPROGRAMM
110 !
120 !
130 ! **********************
140 ! *                    *
150 ! *                    *
160 ! * PROGRAMMIERT VON:  *
170 ! * MICHAEL GARTNER    *
180 ! * STERNSTR.7         *
190 ! * 8063 ODELZHAUSEN   *
200 ! *                    *
210 ! * 24.02.1983         *
220 ! *                    *
230 ! **********************
240 !
250 !
260 CALL CLEAR :: CALL SCREEN(2)
270 DIM F$(18),P$(7),Q$(12),GS$(26),KS$(26),ZS$(10),TA(18),MO(18),Q(12)
280 FOR @=1 TO 18 :: READ F$(@):: NEXT @ :: FOR @=1 TO 7 :: READ P$(@):: NEXT @
290 FOR @=1 TO 12 :: READ Q$(@):: NEXT @ :: FOR @=1 TO 12 :: READ Q(@):: NEXT @
300 FOR @=1 TO 26 :: READ GS$(@):: CALL CHAR(64+@,GS$(@)):: NEXT @
310 FOR @=1 TO 26 :: READ KS$(@):: CALL CHAR(96+@,KS$(@)):: NEXT @
320 FOR @=1 TO 10 :: READ ZS$(@):: CALL CHAR(47+@,ZS$(@)):: NEXT @
330 FOR @=48 TO 57 :: CALL CHARPAT(@,C$):: CALL CHAR(@+80,C$):: NEXT @
340 CALL CHARPAT(83,K1$):: CALL CHARPAT(111,K2$):: CALL CHAR(138,K1$):: CALL CHA
R(139,K2$)
350 CALL CHAR(46,"0000000000030"):: CALL CHAR(58,"000003000030"):: CALL CHAR(
60,"0000FF")
360 CALL CHAR(36,"0101010101010101"):: CALL CHAR(37,"8080808080808080"):: CALL C
HAR(38,"00000000000000FF"):: CALL CHAR(39,"FF")
370 CALL CLEAR :: CALL SCREEN(16):: FOR @=0 TO 12 :: CALL COLOR(@,16,5):: NEXT @
 :: CALL COLOR(13,7,16,14,7,16):: CALL RAHMEN
380 CALL HCHAR(6,4,38,26):: CALL HCHAR(11,4,38,26):: CALL HCHAR(20,4,39,26):: CA
LL VCHAR(7,4,37,13):: CALL VCHAR(7,29,36,13)
390 DISPLAY AT(4,7):"KALENDERPROGRAMM" :: DISPLAY AT(8,6)SIZE(18):"Anwendungsber
eich:" :: DISPLAY AT(10,6)SIZE(18):"von 1583 bis 2200"
400 DISPLAY AT(13,7)SIZE(15):"Heutiges Datum:"
410 DISPLAY AT(15,8)SIZE(14):"Eingabeformat:" :: DISPLAY AT(16,9)SIZE(12):"(01.0
1.1982)"
```

```
420 ACCEPT AT(18,10)VALIDATE(DIGIT,".")SIZE(10)BEEP:V$ :: IF V$="" THEN 420 :: I
F LEN(V$)<10 THEN 420 :: T=VAL(SEG$(V$,1,2)):: M=VAL(SEG$(V$,4,2)):: JZ=VAL(SEG$
(V$,7,4))
430 ON ERROR 420 :: GOSUB 2110 :: IF Y0=1 THEN 420 :: GOSUB 2000 :: HT=TX
440 CALL CLEAR :: CALL SCREEN(7):: FOR @=0 TO 12 :: CALL COLOR(@,2,16):: NEXT @
:: CALL RAHMEN :: DISPLAY AT(3,7):"KALENDERPROGRAMM"
450 CALL HCHAR(4,3,38,28):: CALL HCHAR(8,3,39,28):: CALL VCHAR(5,3,37,3):: CALL
VCHAR(5,30,36,3)
460 DISPLAY AT(6,9)SIZE(12):"Auswahlliste"
470 DISPLAY AT(10,2):"(1) = Monatskalender" :: DISPLAY AT(12,2):"(2) = Feiertage
 im Jahr"
480 DISPLAY AT(14,2):"(3) = Wochentage" :: DISPLAY AT(16,2):"(4) = Distanz" :: D
ISPLAY AT(18,2):"(5) = Programm beenden"
490 DISPLAY AT(22,2):"Teilprogramm:" :: ACCEPT AT(22,16)VALIDATE("12345")SIZE(1)
BEEP:EI$ :: IF EI$="" THEN 490
500 EI=VAL(EI$):: ON EI GOTO 510,990,1170,1330,1570
510 CALL CLEAR :: DISPLAY AT(2,7):"MONATSKALENDER"
520 DISPLAY AT(4,1):"Monat:";TAB(18);"Jahr:" :: CALL HCHAR(5,2,60,29)
530 ACCEPT AT(4,8)VALIDATE(UALPHA)SIZE(9)BEEP:M$ :: IF M$="" THEN 530
540 FOR M1=1 TO 12
550 IF Q$(M1)=M$ THEN 580
560 NEXT M1
570 GOTO 530
580 ACCEPT AT(4,24)VALIDATE(DIGIT)SIZE(4)BEEP:JZ$ :: IF JZ$="" THEN 580 ELSE JZ=
VAL(JZ$)
590 IF JZ>2200 OR JZ<1583 THEN 580
600 DISPLAY AT(7,4):CHR$(138)&CHR$(139)&" Mo Di Mi Do Fr Sa"
610 CALL HCHAR(8,5,60,22)
620 DISPLAY AT(17,1):"Feiertage und kirchl.Feste" :: N=1
630 CALL HCHAR(18,2,60,29)
640 GOSUB 1580 :: M=M1 :: T=1 :: GOSUB 2000
650 R=TX-INT(TX/7)*7 :: IF JZ/4=INT(JZ/4)AND JZ/100<>INT(JZ/100)OR JZ/400=INT(JZ
/400)THEN Q(2)=29
660 V1=Q(M):: X=1 :: L=10
670 FOR R=R TO 6 :: GOSUB 840 :: X=X+1 :: NEXT R
680 FOR L=11 TO 16 :: FOR R=0 TO 6 :: GOSUB 840 :: IF X=V1 THEN 700
690 X=X+1 :: NEXT R :: NEXT L
700 FOR I=1 TO 18
710 MA$=STR$(MO(I)):: TA$=STR$(TA(I))
720 IF LEN(MA$)=2 THEN 740
730 MA$="0"&MA$
740 IF LEN(TA$)=2 THEN 760
750 TA$="0"&TA$
760 IF MO(I)=M THEN DISPLAY AT(18+N,1):F$(I);TAB(24);TA$;".";MA$ :: N=N+1
770 NEXT I
780 FOR I=1 TO 18
790 IF MO(I)=M THEN 820
800 NEXT I
810 DISPLAY AT(20,4):"im ";M$;JZ :: DISPLAY AT(22,4):"sind keine Feiertage"
820 CALL KEY(0,L1,L2):: IF L2=0 THEN 820
830 IF L1=56 THEN 510 ELSE IF L1=57 THEN 440 ELSE 820
840 X$=STR$(X):: LX=LEN(X$)
850 IF LX=2 THEN 870
860 X$=" "&X$
870 IF R=0 THEN 930
880 FOR I=1 TO 18
890 IF MO(I)=M AND TA(I)=X THEN 930
900 NEXT I
910 DISPLAY AT(L,(R*3)+4):X$
920 GOTO 980
930 A=ASC(SEG$(X$,1,1)):: B=ASC(SEG$(X$,2,1))
940 IF A=32 THEN 960
950 A=A+80
960 B=B+80
970 DISPLAY AT(L,(R*3)+4):CHR$(A);CHR$(B)
980 RETURN
990 CALL CLEAR :: DISPLAY AT(2,6):"FEIERTAGE IM JAHR"
1000 DISPLAY AT(4,1):"Feiertage fuer das Jahr"
1010 CALL HCHAR(5,2,60,30)
1020 ACCEPT AT(4,25)VALIDATE(DIGIT)SIZE(4)BEEP:JZ$ :: IF JZ$="" THEN 1020 ELSE J
Z=VAL(JZ$)
1030 IF JZ>2200 OR JZ<1583 THEN 1020
1040 GOSUB 1580
1050 FOR I=1 TO 18
1060 MO$=STR$(MO(I)):: TA$=STR$(TA(I))
1070 IF LEN(MO$)=2 THEN 1090
```

```
1080 MO$="0"&MO$
1090 IF LEN(TA$)=2 THEN 1110
1100 TA$="0"&TA$
1110 T=TA(I):: M=MO(I)
1120 GOSUB 2000 :: AS=TX-INT(TX/7)*7
1130 DISPLAY AT(5+I,1):F$(I);TAB(21);TA$;".";MO$;" ";SEG$(P$(AS+1),1,2)
1140 NEXT I
1150 CALL KEY(0,L1,L2):: IF L2=0 THEN 1150
1160 IF L1=56 THEN 990 ELSE IF L1=57 THEN 440 ELSE 1150
1170 CALL CLEAR :: CALL RAHMEN :: DISPLAY AT(5,9):"WOCHENTAGE"
1180 DISPLAY AT(9,7):"Eingabeformat:" :: DISPLAY AT(10,8):"(01.01.1982)" :: DISP
LAY AT(13,6):"Datum:"
1190 ACCEPT AT(13,13)VALIDATE(DIGIT,".")SIZE(10)BEEP:V$
1200 IF V$="" THEN 1190
1210 TA$=SEG$(V$,1,2):: MO$=SEG$(V$,4,2):: J$=SEG$(V$,7,4)
1220 ON ERROR 1190 :: T=VAL(TA$):: M=VAL(MO$):: JZ=VAL(J$)
1230 GOSUB 2110 :: IF Y0=1 THEN 1190
1240 IF JZ>2200 OR JZ<1583 THEN 1190
1250 GOSUB 2000 :: AS=TX-INT(TX/7)*7
1260 IF TX>=HT THEN HT$="ist" ELSE HT$="war"
1270 DISPLAY AT(9,6):"                    " :: DISPLAY AT(10,8):"            " ::
DISPLAY AT(13,6):"                "
1280 CALL HCHAR(9,4,38,26):: CALL HCHAR(18,4,39,26):: CALL VCHAR(10,4,37,8):: CA
LL VCHAR(10,29,36,8)
1290 DISPLAY AT(12,4)SIZE(22):"Der ";V$;" ";HT$;" ein"
1300 DISPLAY AT(15,10)SIZE(10):P$(AS+1)
1310 CALL KEY(0,L1,L2):: IF L2=0 THEN 1310
1320 IF L1=56 THEN 1170 ELSE IF L1=57 THEN 440 ELSE 1310
1330 CALL CLEAR :: CALL RAHMEN :: DISPLAY AT(5,10):"DISTANZ"
1340 DISPLAY AT(9,7):"Eingabeformat:" :: DISPLAY AT(10,8):"(01.01.1982)"
1350 DISPLAY AT(13,4):"Erstes Datum:" :: ACCEPT AT(13,18)VALIDATE(DIGIT,".")SIZE
(10)BEEP:V$ :: V1$=V$
1360 IF V$="" THEN 1350
1370 TA$=SEG$(V$,1,2):: MO$=SEG$(V$,4,2):: J$=SEG$(V$,7,4)
1380 ON ERROR 1350 :: T=VAL(TA$):: M=VAL(MO$):: JZ=VAL(J$)
1390 GOSUB 2110 :: IF Y0=1 THEN 1350
1400 IF JZ>2200 OR JZ<1583 THEN 1350
1410 GOSUB 2000 :: TX1=TX
1420 DISPLAY AT(15,3):"Zweites Datum:" :: ACCEPT AT(15,18)VALIDATE(DIGIT,".")SIZ
E(10)BEEP:V$
1430 IF V$="" THEN 1420
1440 TA$=SEG$(V$,1,2):: MO$=SEG$(V$,4,2):: J$=SEG$(V$,7,4)
1450 ON ERROR 1420 :: T=VAL(TA$):: M=VAL(MO$):: JZ=VAL(J$)
1460 GOSUB 2110 :: IF Y0=1 THEN 1420
1470 IF JZ>2200 OR JZ<1583 THEN 1420
1480 GOSUB 2000 :: TX2=TX
1490 DISPLAY AT(9,6):"                    " :: DISPLAY AT(10,8):"            " :: D
ISPLAY AT(13,3):"                    "
1500 DISPLAY AT(15,3):"                    "
1510 CALL HCHAR(9,3,38,28):: CALL HCHAR(19,3,39,28):: CALL VCHAR(10,3,37,9):: CA
LL VCHAR(10,30,36,9)
1520 DISPLAY AT(12,9)SIZE(14):"vom ";V1$ :: DISPLAY AT(14,5)SIZE(18):"bis zum ";
V$
1530 IF TX2-TX1=1 THEN DISPLAY AT(16,5)SIZE(13):" ist es 1 Tag"
1540 IF TX2-TX1<>1 THEN DISPLAY AT(16,5)SIZE(19):"sind es";TX2-TX1;"Tage"
1550 CALL KEY(0,L1,L2):: IF L2=0 THEN 1550
1560 IF L1=56 THEN 1330 ELSE IF L1=57 THEN 440 ELSE 1550
1570 CALL CLEAR :: END
1580 MM=22 :: NN=2 :: M=3
1590 IF JZ>=1700 THEN MM=MM+1
1600 IF JZ>=1900 THEN MM=MM+1
1610 IF JZ>=2200 THEN MM=MM+1
1620 IF MM=23 THEN NN=NN+1
1630 IF MM=23 AND JZ>=1800 THEN NN=NN+1
1640 IF MM=24 THEN NN=NN+3
1650 IF MM=24 AND JZ>=2100 THEN NN=NN+1
1660 IF MM=25 THEN NN=0
1670 TB=((JZ-INT(JZ/19)*19)*19+MM)-INT(((JZ-INT(JZ/19)*19)*19+MM)/30)*30
1680 TC=TB*6+(JZ-INT(JZ/7)*7)*4+(JZ-INT(JZ/4)*4)*2+NN
1690 WO=TC-INT(TC/7)*7 :: TX=TB+WO
1700 IF TX=35 THEN TX=28
1710 IF TX=34 AND TB=28 AND(JZ-INT(JZ/19)*19)>10 THEN TX=27
1720 TX=TX+22
1730 IF TX>31 THEN M=M+1 :: TX=TX-31
1740 TA(4)=TX :: MO(4)=M
1750 TA(3)=TA(4)-2 :: MO(3)=MO(4)
```

```
1760 IF TA(3)<1 THEN MO(3)=MO(3)-1 :: TA(3)=TA(3)+31
1770 TA(5)=TA(4)+1 :: MO(5)=MO(4)
1780 IF TA(5)=32 THEN MO(5)=MO(5)+1 :: TA(5)=TA(5)-31
1790 TA(7)=TA(4)+9 :: MO(7)=5
1800 IF MO(4)=3 THEN MO(7)=4
1810 IF TA(7)>31 THEN MO(7)=MO(7)+1 :: TA(7)=TA(7)-31
1820 TA(8)=TA(4)+19 :: MO(8)=5
1830 IF MO(4)=3 THEN MO(8)=4
1840 IF TA(8)>31 THEN MO(8)=MO(8)+1 :: TA(8)=TA(8)-31
1850 TA(9)=TA(4)+20 :: MO(9)=5
1860 IF MO(4)=3 THEN MO(9)=4
1870 IF TA(9)>31 THEN MO(9)=MO(9)+1 :: TA(9)=TA(9)-31
1880 TA(10)=TA(4)+30 :: MO(10)=5
1890 IF MO(4)=3 THEN MO(10)=4
1900 IF TA(10)>31 THEN MO(10)=MO(10)+1 :: TA(10)=TA(10)-31
1910 T=1 :: M=11 :: GOSUB 2000
1920 WO=TX-INT(TX/7)*7 :: TA(14)=8-WO
1930 IF WO<4 THEN TA(14)=TA(14)-7
1940 TA(14)=TA(14)+14 :: MO(14)=11
1950 TA(15)=TA(14)+3 :: MO(15)=11
1960 TA(16)=TA(14)+7 :: MO(16)=11
1970 TA(1),TA(6),TA(13),MO(1),MO(2)=1 :: TA(2),MO(11)=6 :: MO(17),MO(18)=12
1980 MO(6)=5 :: TA(12)=15 :: MO(12)=8 :: MO(13)=11 :: TA(11)=17 :: TA(17)=25 ::
TA(18)=26
1990 RETURN
2000 Z=0 :: IF M<=2 THEN 2070
2010 IF JZ/4<>INT(JZ/4)THEN 2050
2020 IF JZ/100<>INT(JZ/100)THEN 2040
2030 IF JZ/400<>INT(JZ/400)THEN 2050
2040 Z=1
2050 V=INT((306*M-324)/10)
2060 GOTO 2080
2070 V=(M-1)*31
2080 Z=Z+(JZ-1)*365+INT((JZ-1)/4)
2090 Z=Z-INT((JZ-1)/100)+INT((JZ-1)/400)
2100 TX=Z+V+T :: RETURN
2110 Y0=0 :: IF T>31 OR T<1 OR M>12 OR M<1 THEN Y0=1
2120 IF T>30 AND(M=2 OR M=4 OR M=6 OR M=9 OR M=11)THEN Y0=1
2130 IF T>29 AND M=2 THEN Y0=1
2140 IF T=29 AND M=2 AND INT(JZ/4)<>JZ/4 THEN Y0=1
2150 IF T=29 AND M=2 AND JZ/100=INT(JZ/100)THEN Y0=1
2160 IF T=29 AND M=2 AND INT(JZ/400)=JZ/400 THEN Y0=0
2170 RETURN
2180 DATA Neujahr,Hl.Drei Koenige,Karfreitag,Ostersonntag,Ostermontag,Tag der Ar
beit
2190 DATA Christi Himmelfahrt,Pfingstsonntag,Pfingstmontag,Fronleichnam,T.d.Deut
sch.Einheit,Maria Himmelfahrt
2200 DATA Allerheiligen,Volkstrauertag,Buss- und Bettag,Totensonntag,1.Weihnacht
stag,2.Weihnachtstag
2210 DATA Sonntag,Montag,Dienstag,Mittwoch,Donnerstag,Freitag,Samstag
2220 DATA JANUAR,FEBRUAR,MAERZ,APRIL,MAI,JUNI,JULI,AUGUST,SEPTEMBER,OKTOBER,NOVE
MBER,DEZEMBER
2230 DATA 31,28,31,30,31,30,31,31,30,31,30,31
2240 DATA 003844447C4444,00784478444478,00384440404438,00782424242478,007C407840
407C,007C4078404040
2250 DATA 003844405C4438,0044447C444444,007C101010107C,007C0404044438,0048506050
4844,00404040404007C
2260 DATA 00446C54444444,004464544C4444,00384444444438,00784444784040,0038444454
4C38,00784444784844,00384038044438
2270 DATA 007C1010101010,00444444444438,00424242422418,00424242425A24,0042241818
2442,0041221408080B,007E040810207E
2280 DATA 000038043C443E,00404078444478,00003840404038,0004043C44443C,0000384478
403C,001C2078202020,00003C44443C0438
2290 DATA 00404078444444,00100030101038,00080018080B2810,00202428302824,00202020
20201C,00006854545454,00007844444444
2300 DATA 00003844444438,00007844447840 40,00003C44443C0404,00002C30202020,000038
40380478,00207820202018,00004444444438,00004444442810
2310 DATA 00004444545428,00004428102844,000044444C340438,00007C0810207C
2320 DATA 00384C54546438,00018080808 1C,00384404384 07C,00384418044438,0008182848
7C08
2330 DATA 007C4078044438,00384078444438,007C0408102020,00384438444438,0038444443C
0438
2340 SUB RAHMEN
2350 CALL HCHAR(1,2,38,30):: CALL HCHAR(24,2,39,30):: CALL VCHAR(2,2,37,22):: CA
LL VCHAR(2,31,36,22)
2360 SUBEND
```

Super Kong

Die Spielidee für Super Kong wurde von dem wohl allseits bekannten Donkey Kong abgeleitet. Nach dem Laden des Programmes erscheint ein grafisch, wirklich toller, Bildschirmaufbau. Um spielen zu können, wird außer dem TI-99/4A natürlich, ein Joystick benötigt.
Hierauf wird auch gleich eine Programmbeschreibung gegeben, in der es heißt: Versuche einen kleinen Mann durch ein Gerüst zu steuern, und paß auf, daß er nicht von den Geistern gefressen wird, die von dem Kong gesteuert werden. Hüte dich vor den blauen Fallgruben, es wäre Dein Verderben. Nach Drücken einer Taste kann es jetzt losgehen.
Also dann viel Spaß!

```
100 REM ********************
110 REM *                  *         *  SUPER <> KONG  *          *
  *
120 REM *  PETER JASSEN    *         *                 *          *DU.DEN 11.3.198
4f*          *                   *
130 REM ********************
140 !
150 ! ZEICHENDEFINIETIONEN
160 !
170 CALL CLEAR :: CALL MAGNIFY(3):: CALL SCREEN(2)
180 !   SUPER KONG
190 CALL CHAR(96,"0000000070E1F373B3D3F3F1F1F0F030000000003860C961238F9FBFFFCFF
FF")
200 CALL CHAR(100,"00000000000000000000000000000007F1F0F0F1F3F7F7F7F7F7F3F3F1F7
FFE")
210 CALL CHAR(104,"00000000F8EC06AD0943F3FBFF07FFFF0000000000000000000000080C0E0F
0F8")
220 CALL CHAR(108,"FFFFFBF5EFEDF3FFFFFFFF8707070303F8B878F0E0C0C0E0F0F0F0F0E0F
8FC")
230 !
240 !   MANN RECHTS
250 !
260 CALL CHAR(112,"0307070E0703018367BE0C0E07030407B0F8C070E0C0B0C0C0E0720CF2E04
070")
270 CALL CHAR(116,"0307070E07030103070F0C2A1723020380F8C070E0C0B0C0C0E4F8F4F0E02
0B8")
280 !
290 !   MANN LINKS
300 !
310 CALL CHAR(120,"011F030E0703010303074E304F07020EC0E0E070E0C0B0C1E67D30F0E0C02
0E0")
320 CALL CHAR(124,"011F030E0703010303271F2F0F07041DC0E0E070E0C0B0C0E0F03054E8C44
0C0")
330 !
340 !   MANN TOT
350 !
360 CALL CHAR(92,"000003000000BA8484DC3EBFBBB3FF178080E6B9868080820A0A1ABEF7F7BF
1E")
370 !
380 !   GEISTER (1)&(2)
390 !
400 CALL CHAR(36,"00000007QF1F17131F12181F1F1F1F140000080C0E06020E0A020E0E0E0E0
A0")
410 CALL CHAR(40,"003A45934539030F0F1617170F07461E0056A2C9A29CC0F0F068E8E8F0E064
78")
420 !
430 !   FRAU
440 !
450 CALL CHAR(60,"00000000000003070F1E373F3363676E2D0A1F0E04008CC0E078E0C28C1AB07A
8C")
460 !
470 ! FLAME (1)&(2)
480 !
490 CALL CHAR(128,"0000000000000103071C1E3F3F301F0F0700703C3EFEFFFFFFCEECF8F078FCF
```

```
CF0")
500 CALL CHAR(136,"000E3C7C7FFFFFFF73371F0F1E3F3F0F0000000000080C0E03878FCFC0CF8F
0E0")
510 !
520 ! LEITER
530 !
540 CALL CHAR(133,"C3C3C3FFFFC3C3C3")
550 !
560 !    GERUEST
570 !
580 CALL CHAR(140,"FFFFF0783C1E1FFFF"):: CALL CHAR(58,"00FFFFFFFFFFFFFF")
590 !
600 !   ZEISCHEN DATAS
610 !
620 DATA 7CFECEDEF6E6FE7C,1838787818181818,7CFE0E3C78E0FEFE,7CFE0E3C3C0EFE7C,1C3
C7CECFEFE0C0C,FEFEC0FCFE06FE7C
630 DATA 7CFEC0FCFEC6FE7C,FEFE060E1C383030,7CFEC67C7CC6FE7C,7CFEC6FE7E06FE7C
640 FOR I=1 TO 10
650 READ A$ :: CALL CHAR(47+I,A$)
660 NEXT I
670 DATA 387CEEC6FEFEC6C6,FCFEC6FCFCC6FEFC,7CFEC0C0C0C0FE7C,FCFEC6C6C6C6FEFC,FEF
EC0F8F8C0FEFE,FEFEC0F8F8C0C0C0,7CFEC0CECEC6FE7C,C6C6C6FEFEC6C6C6
680 DATA 1818181818181818,FEFE06060606FE7C,C6CEDCF8F8DCCEC6,C0C0C0C0C0C0FEFE,C6E
EFEFED6C6C6C6,C6E6E6F6DECECEC6,7CFEC6C6C6C6FE7C,FCFEC6FEFCC0C0C0
690 DATA 7CFEC6C6D6CCFE76,FCFEC6FEFCDCCEC6,7CFEC0FC7E06FE7C,FEFE1818181818,C6C
6C6C6C6C6FE7C,C6C6EE6C7C38381,C6C6C6C6D6FEFE6C,C6EE7C38387CEEC6
700 DATA C6CEFC783030303,FEFE0E1C3870FEFE
710 FOR I=1 TO 26
720 READ A$
730 CALL CHAR(64+I,A$)
740 NEXT I
750 !
760 !   ANFANGSBILD
770 !
780 DATA 3,1,8,3,25,8,6,1,32,11,1,7,11,12,10,11,26,7,15,7,20
790 DATA 4,8,2,4,25,2,7,4,4,7,12,4,7,21,4,7,29,4,12,7,3,12,26,3
800 FOR I=1 TO 7
810 READ ZI,SP,L
820 CALL HCHAR(ZI,SP,140,L)
830 NEXT I
840 FOR I=1 TO 8
850 READ ZI,SP,L
860 CALL VCHAR(ZI,SP,133,L)
870 NEXT I
880 !
890 !     TEXT
900 !
910 DISPLAY AT(19,1)SIZE(5):"SUPER" :: DISPLAY AT(1,7):"SUPER"
920 DISPLAY AT(20,25)SIZE(4):"KONG" :: DISPLAY AT(1,19):"KONG"
930 DISPLAY AT(17,10):"PETER JANSSEN"
940 DISPLAY AT(19,10)SIZE(8):"SOFTWARE"
950 DISPLAY AT(21,10):"PRESENTIRT"
960 DISPLAY AT(23,5):"SPIELREGELN ERWUENSCHT J N"
970 !
980 !   SPRITE
990 !
1000 CALL SPRITE(#1,36,2,1,24,#2,40,2,25,72,#3,40,2,25,168,#4,36,2,1,216)
1010 CALL SPRITE(#5,96,2,8,112,#6,100,2,24,112,#7,104,2,8,128,#8,108,2,24,128)
1020 CALL SPRITE(#9,36,2,64,40,#10,92,2,64,120,#11,36,2,64,200)
1030 CALL SPRITE(#12,112,2,97,72,#13,40,2,97,120,#14,124,2,97,168)
1040 !
1050 !   COLOR
1060 !
1070 CALL COLOR(13,16,2,14,14,2)
1080 CALL COLOR(#1,13,#2,11,#3,11,#4,13)
1090 CALL COLOR(#5,9,#6,9,#7,9,#8,9)
1100 CALL COLOR(#9,13,#10,6,#11,13)
1110 CALL COLOR(#12,8,#13,12,#14,8)
1120 !
1130 ! ABFRAGE OB J N
1140 !
1150 FOR I=2 TO 10 :: RANDOMIZE :: CALL COLOR(I,INT(RND*13)+3,2):: NEXT I
1160 FOR I=1 TO 300 :: NEXT I
1170 CALL MOTION(#2,0,7,#3,0,-7)
1180 CALL KEY(0,K,S)
1190 IF K=74 OR K=106 THEN 1300
1200 IF K=78 OR K=110 THEN 1560
1210 CALL MOTION(#10,-3,0)
1220 FOR L=1 TO 100 :: NEXT L
1230 CALL MOTION(#10,3,0)
1240 FOR L=1 TO 100 :: NEXT L
1250 CALL MOTION(#10,0,0)
1260 GOTO 1180
1270 !
1280 ! SPIELERKLERUNG
1290 !
1300 CALL CLEAR :: CALL DELSPRITE(ALL)
1310 DISPLAY AT(1,10):"SUPER KONG"
1320 DISPLAY AT(4,1):"MAN MUSS VERSUCHEN EINEN KL"
1330 DISPLAY AT(6,1):"EINEN MAN DURCH EIN GERUEST"
1340 DISPLAY AT(8,1):"ZU STEUERN DABEI MUST DU AUF "
```

```
1350 DISPLAY AT(10,1):"PASSEN DAS DU NICHT VON DEN"
1360 DISPLAY AT(12,1):"GEISTERN GEFRESSEN WIRST DIE"
1370 DISPLAY AT(14,1):"DER KONG STEUERT HUETE DICH"
1380 DISPLAY AT(16,1):"VOR DEN BLAUEN FALLGRUBEN"
1390 DISPLAY AT(18,1):"ES WAERE DEIN VERDERBEN "
1400 DISPLAY AT(22,4):"DRUECKE EINE TASTE"
1410 CALL KEY(0,K,S):: IF S=0 THEN 1410
1420 CALL CLEAR
1430 DISPLAY AT(1,10):"SUPER KONG"
1440 DISPLAY AT(4,1):"SOLTEST DU ES SCHAFFEN ZUR "
1450 DISPLAY AT(6,1):"FRAU ZU GELANGEN IST DIE "
1460 DISPLAY AT(8,1):"ERSTE LAGE GESCHAFFT SOMIT "
1470 DISPLAY AT(10,1):"ERHOET SICH DIE SCHWIRICKEIT"
1480 DISPLAY AT(15,3):"LENCKEN MIT JOYSTIGS"
1490 DISPLAY AT(17,8):"VIEL SPASS"
1500 DISPLAY AT(20,4):"SPIELBEGIN MIT ENTER "
1510 CALL KEY(0,K,S)
1520 IF S=1 THEN 1560 ELSE 1510
1530 !
1540 !   SPIEL SUPER KONG
1550 !
1560 CALL CLEAR :: CALL DELSPRITE(ALL):: RESTORE 1630
1570 CALL COLOR(13,2,2,14,2,2,4,2,2)
1580 !
1590 !  FELTAUFBAU
1600 !
1610 CALL HCHAR(4,3,140,6)
1620 FOR I=3 TO 20 STEP 5 :: CALL HCHAR(25-I,1,140,32):: NEXT I
1630 DATA 7,12,12,8,12,22,17,15
1640 DATA 8,17,13,5,13,28,18,12,18,20
1650 FOR I=1 TO 4
1660 READ ZI,SP
1670 CALL HCHAR(ZI,SP,58,3)
1680 NEXT I
1690 CALL VCHAR(5,8,133,2)
1700 FOR I=1 TO 5
1710 READ ZI,SP
1720 CALL VCHAR(ZI,SP,133,4)
1730 NEXT I
1740 CALL HCHAR(3,1,58,2):: CALL HCHAR(4,1,58,2)
1750 CALL COLOR(13,16,2,14,14,2,4,6,2)
1760 CALL COLOR(9,10,2,10,10,2)
1770 DISPLAY AT(5,23):"dfln" :: DISPLAY AT(6,23):"egmo" :: CALL SPRITE(#1,96,10,
17,192,#3,104,10,17,208)
1780 CALL SPRITE(#5,40,10,33,144,#6,40,4,72,40,#7,40,4,72,180,#8,36,12,112,40)
1790 CALL SPRITE(#9,40,5,112,120,#10,36,12,112,200,#11,36,15,152,104,#12,36,15,1
52,200)
1800 CALL SPRITE(#13,60,7,1,5)
1810 DISPLAY AT(2,10):"SUPER KONG"
1820 DISPLAY AT(24,1)SIZE(6):"PUNKTE" :: DISPLAY AT(23,8)SIZE(6):"BONUS"
1830 DISPLAY AT(23,22)SIZE(6):"prprpr" :: DISPLAY AT(24,22)SIZE(6):"qsqsqs"
1840 CALL COLOR(11,8,2)
1850 FOR I=1 TO 400 :: NEXT I
1860 CALL MOTION(#5,0,14,#6,0,-10,#7,0,-10,#8,0,7,#10,0,7,#9,0,7,#11,0,-6,#12,0,
-6)
1870 DISPLAY AT(24,7)SIZE(7):" 0    "
1880 !
1890 !   HAUPTPROGRAMM***
1900 !
1910 DISPLAY AT(23,13)SIZE(7):" 0    "
1920 WA=1 :: SPALTE=0 :: APD=0 :: X=0 :: Y=0 :: FG=112 :: GF=116 :: KL=0 :: BONU
S=0 :: GSC=0
1930 CALL SPRITE(#14,112,8,152,8,0,0)
1940 CALL JOYST(1,X,Y):: CALL MOTION(#14,0,X*1.5):: IF X=-4 THEN FG=120 :: GF=12
4 :: KL=-18 ELSE IF X=4 THEN FG=112 :: GF=116 :: KL=18
1950 CALL PATTERN(#14,FG)
1960 ON WA GOTO 2250,2290,2340,2390
1970 !
1980 !  SPRUNG
1990 !
2000 CALL PATTERN(#14,GF):: CALL SOUND(-1,120,0)
2010 CALL KEY(1,K,S):: IF K=18 THEN 2020 ELSE 1940
2020 CALL MOTION(#14,-24,KL)
2030 FOR I=1 TO 3 :: CALL COINC(ALL,Q):: IF Q=-1 THEN 2440 ELSE CALL SOUND(50,40
0*I,0)
2040 NEXT I
2050 CALL MOTION(#14,24,KL)
2060 FOR I=1 TO 3 :: CALL COINC(ALL,Q):: IF Q=-1 THEN 2440 ELSE CALL SOUND(50,16
00/I,0)
2070 NEXT I
2080 CALL POSITION(#14,A,B):: CALL MOTION(#14,0,0)
2090 CALL LOCATE(#14,SPALTE,B)
2100 BONUS=BONUS+50 :: DISPLAY AT(23,13)SIZE(7):BONUS :: PU=PU+5 :: DISPLAY AT(2
4,7)SIZE(7):PU
2110 CALL COINC(ALL,Q):: IF Q=-1 THEN 2440 ELSE 1940
2120 !
2130 ! LEITER
2140 !
2150 CALL JOYST(1,X,Y):: IF Y=-4 THEN 2160 ELSE CALL MOTION(#14,-Y*4,0)
2160 CALL COINC(ALL,Q):: IF Q=-1 THEN 2440
2170 CALL POSITION(#14,A,B):: IF A<APD+8 THEN 2180 ELSE 2150
```

```
2180 CALL MOTION(#14,0,0)
2190 CALL LOCATE(#14,APD,B)
2200 BONUS=BONUS+100 :: DISPLAY AT(23,13)SIZE(7):BONUS :: PU=PU+INT(RND*30)+10 :
: DISPLAY AT(24,7)SIZE(7):PU
2210 IF WA>3 THEN 1940 ELSE WA=WA+1 :: GOTO 1940
2220 !
2230 ! ABFRAGEN
2240 !
2250 SPALTE=152 :: APD=112
2260 CALL DISTANCE(#14,152,85,Q):: IF Q<10 AND Y=4 THEN 2150
2270 CALL DISTANCE(#14,152,149,Q):: IF Q<10 AND Y=4 THEN 2150
2280 CALL COINC(ALL,Q):: IF Q=-1 THEN 2440 ELSE 2000
2290 SPALTE=112 :: APD=72
2300 CALL DISTANCE(#14,112,113,Q2):: IF Q2<120 THEN 2590
2310 CALL DISTANCE(#14,112,29,Q):: IF Q<10 AND Y=4 THEN 2150
2320 CALL DISTANCE(#14,112,213,Q):: IF Q<10 AND Y=4 THEN 2150
2330 CALL COINC(ALL,Q):: IF Q=-1 THEN 2440 ELSE 2000
2340 SPALTE=72 :: APD=32
2350 CALL DISTANCE(#14,72,57,Q3):: IF Q3<120 THEN 2590
2360 CALL DISTANCE(#14,72,168,Q3):: IF Q3<120 THEN 2590
2370 CALL DISTANCE(#14,72,126,Q):: IF Q<10 AND Y=4 THEN 2150
2380 CALL COINC(ALL,Q):: IF Q=-1 THEN 2440 ELSE 2000
2390 SPALTE=32
2400 CALL DISTANCE(#14,32,89,Q4):: IF Q4<120 THEN 2590
2410 CALL DISTANCE(#14,32,54,Q):: IF Q<10 AND Y=4 THEN 2680
2420 CALL COINC(ALL,Q):: IF Q=-1 THEN 2440 ELSE 2000
2430 !
2440 !   TOT
2450 !
2460 CALL PATTERN(#14,92):: CALL MOTION(#14,0,0)
2470 CALL MOTION(#14,-3,4)
2480 FOR I=1 TO 2
2490 CALL SOUND(500,660,0,670,0,680,0)
2500 CALL SOUND(500,560,0,570,0,560,0)
2510 CALL SOUND(500,680,0,670,0,650,0)
2520 CALL SOUND(500,580,0,570,0,560,0)
2530 NEXT I
2540 MAN=MAN+2
2550 DISPLAY AT(23,20+MAN)SIZE(2):"  " :: DISPLAY AT(24,20+MAN)SIZE(2):"  "
2560 CALL DELSPRITE(#14):: IF MAN=8 THEN 2910 ELSE CALL LOCATE(#11,152,104,#12,1
52,200)
2570 GOTO 1890
2580 !
2590 !   FALLGRUBE
2600 !
2610 CALL MOTION(#14,3,0)
2620 FOR I=1 TO 15
2630 CALL SOUND(200,1000-I,I,990-I,I)
2640 NEXT I
2650 CALL SOUND(800,120,0,220,0):: CALL PATTERN(#14,92)
2660 GOTO 2470
2670 !
2680 !   FRAU GERETTET
2690 !
2700 CALL MOTION(#14,-9,0):: LAGE=LAGE+1
2710 FOR I=1 TO 10
2720 CALL SOUND(50,200*I,0)
2730 NEXT I
2740 CALL MOTION(#14,0,0)
2750 FOR I=1 TO 5
2760 CALL SOUND(200,2000,0,3000,0)
2770 CALL SOUND(200,3000,I,2000,I)
2780 NEXT I
2790 PU=PU+BONUS :: DISPLAY AT(24,7)SIZE(7):PU
2800 ON LAGE GOTO 2830,2870,2810
2810 LAGE=0 :: GOTO 2560
2820 !
2830 !   LAGEN
2840 !
2850 CALL PATTERN(#5,40,#6,128,#7,36,#8,40,#9,139,#10,36,#11,128,#12,40)
2860 CALL COLOR(#5,16,#6,8,#7,10,#8,15,#10,15,#9,3,#11,12,#12,12):: CALL MOTION(
#11,0,-10,#12,0,-10):: GOTO 2560
2870 CALL MOTION(#11,0,-8,#12,0,-8)
2880 CALL PATTERN(#5,136,#6,128,#7,128,#8,136,#9,136,#10,136,#11,128,#12,128)
2890 CALL COLOR(#5,7,#6,4,#7,12,#8,5,#9,10,#10,5,#11,16,#12,16):: GOTO 2560
2900 !
2910 !   END
2920 !
2930 IF PU>HOE THEN HOE=PU
2940 DISPLAY AT(9,3)SIZE(7):"HOECHST" :: DISPLAY AT(9,19):"PU ";HOE
2950 MAN=0 :: PU=0 :: LAGE=0 :: DISPLAY AT(19,2)SIZE(24):"NEUES    SPIEL    J
N"
2960 FOR I=1 TO 20
2970 CALL KEY(0,K,S)
2980 IF K=74 OR K=106 THEN 1560 ELSE IF K=78 OR K=110 THEN CALL CLEAR :: CALL DE
LSPRITE(ALL):: END
2990 NEXT I
3000 DISPLAY AT(14,9)SIZE(10):"          "
3010 FOR I=1 TO 100 :: NEXT I
3020 DISPLAY AT(14,9)SIZE(10):"SPIEL ENDE"
3030 GOTO 2960
```

Donald der Dschungelheld

für den TI-99/4A und Extended Basic

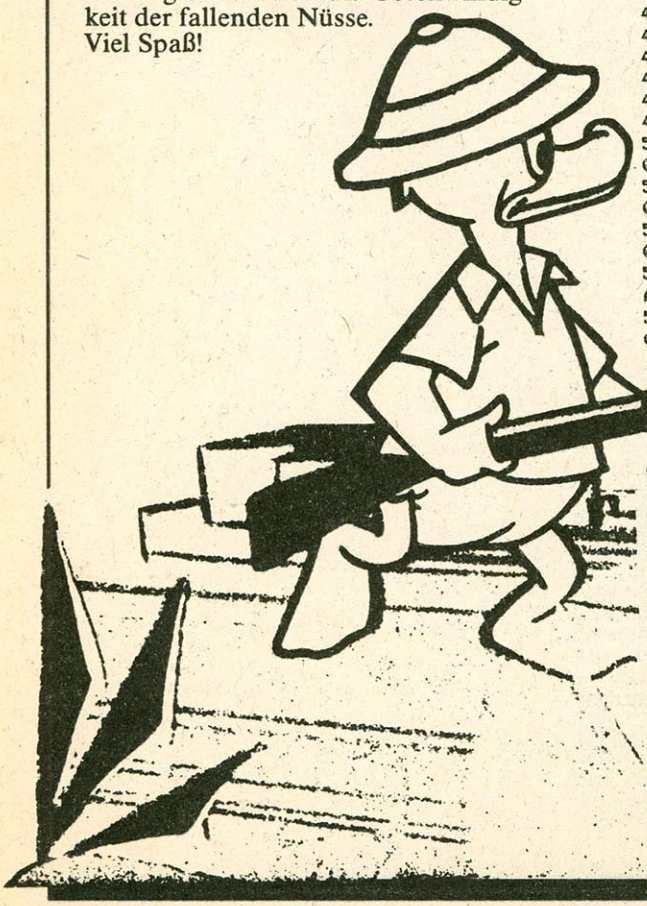
Daisy Duck, von Donald heiß begehrt, wurde von schwarzen Affen gefangengenommen und in einen Käfig gesteckt. Unser Held, Donald, versucht nun, Daisy zu befreien. Hierzu muß er den Käfig seiner Freundin erreichen. Die Affen machen ihm die Sache schwer, indem sie mit übergroßen Kokusnüssen nach Donald werfen. Trifft eine Nuß unseren Helden, hat er sein Leben verwirkt.

Hat Donald es geschafft, die Nüsse heil zu überstehen, erwartet ihn unmittelbar vor dem Käfig eine recht gemeine Schlange, die unerwartet versucht, Donald zu beißen.

Ist endlich der Käfig erreicht, zeigt sich auf dem Bildschirm die befreite Daisy mit ihrem Helden Donald.

Mit fünf Spielfiguren wird gestartet. Bei jedem Verlust wird ein Donald abgezogen. Die Punktwertung erfolgt nach überstandener Gefahr und erfolgreicher Rettung. Für jeden Rettungsversuch, der erfolgreich abgeschlossen wurde, erhält der Spieler 100 Pkt. Zusätzlich erfolgt die Restzeit als positiv auf dem Konto. Bei Nichteinhaltung des Zeitlimits (Time = 50) hat Donald sein Leben ebenfalls verwirkt. Mit jeder Rettung erhöht sich die Geschwindigkeit der fallenden Nüsse.

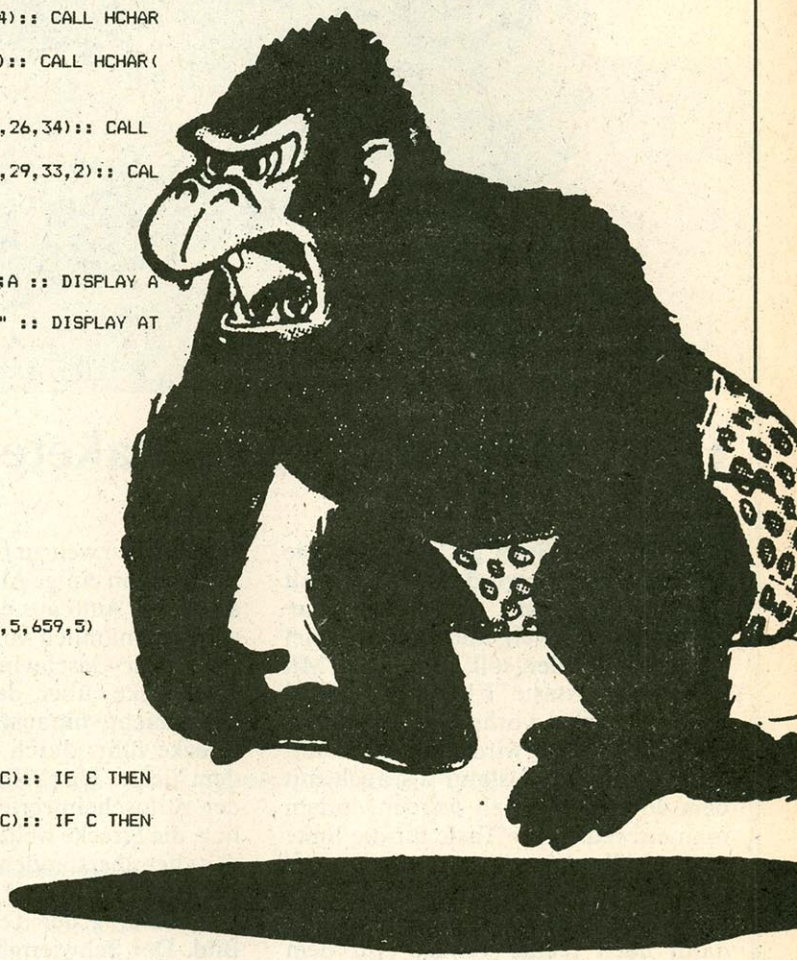
Viel Spaß!

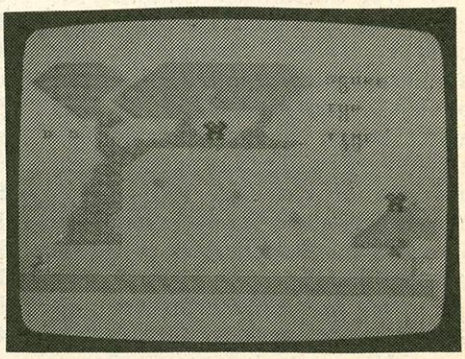


```
10 REM *****************
20 REM * DONALD DER     *
30 REM * DSCHUNGELHELD *
40 REM *               *
50 REM * WOLFGANG KLEIN*
60 REM * HAFENSTR. 12  *
70 REM * 2000 WEDEL(H) *
80 REM *    JUNI 84    *
90 REM *****************
100 T=0
110 RANDOMIZE
120 A=0 :: D=5 :: TI=50 :: GZ=0
130 CALL CLEAR :: CALL SCREEN(12)
140 CALL MAGNIFY(3)
150 CALL CHAR(91,"7E81999191998I7E")
160 CALL CHAR(96,"002321232B272123372B2120202120230000B0C0B00000B0C0E8F8F0A02090
B8")! DONALD NACH LINKS (A)
170 CALL CHAR(100,"000301834B27110B070301000010A040000B0C0B00000B0C0E8F8F0A01C0
404")! DONALD NACH LINKS (B)
180 CALL CHAR(104,"000001030100000103171F0F0504091D00C484C4D4E484C4ECD4840404840
4C4")! DONALD NACH RECHTS (A)
190 CALL CHAR(108,"000001030100000103171F0F0538202000C0B0C1D2E488D0E0C0800000B05
020")! DONALD NACH RECHTS (B)
200 DISPLAY AT(6,4):"DONALD DER DSCHUNGELHELD"
210 DISPLAY AT(17,5):"PRESS ANY KEY TO START"
220 DISPLAY AT(22,8):"[ WOLFGANG KLEIN"
230 CALL SPRITE(#2,104,2,12*8,256)
240 CALL SPRITE(#1,96,2,9*8,256)
250 CALL SOUND(250,880,0):: CALL SOUND(250,783,0):: CALL SOUND(125,659,0):: CALL
 SOUND(175,523,0):: CALL SOUND(125,659,0)
260 CALL SOUND(250,587,0):: CALL SOUND(125,659,0):: CALL SOUND(125,587,0):: CALL
 SOUND(125,587,0):: CALL SOUND(125,783,0)
270 CALL SOUND(125,659,0):: CALL SOUND(125,587,0):: CALL SOUND(125,587,0):: CALL
 SOUND(125,523,0):: CALL SOUND(125,659,0)
280 CALL SOUND(125,587,0):: CALL SOUND(125,587,0):: CALL SOUND(125,783,0):: CALL
 SOUND(125,659,0):: CALL SOUND(125,587,0)
290 CALL SOUND(125,587,0):: CALL SOUND(250,880,0):: CALL SOUND(250,783,0):: CALL
 SOUND(125,659,0):: CALL SOUND(175,523,0)
300 CALL SOUND(125,659,0):: CALL SOUND(500,587,0)
310 CALL MOTION(#1,0,-10,#2,0,10)
320 FOR L=1 TO 5
330 CALL PATTERN(#1,100):: CALL PATTERN(#2,108):: FOR I=1 TO 40 :: NEXT I
340 CALL PATTERN(#1,96):: CALL PATTERN(#2,104):: FOR I=1 TO 40 :: NEXT I
350 NEXT L
360 CALL MOTION(#1,0,0,#2,0,0)
370 CALL KEY(0,K,S):: IF S=0 THEN 370
380 CALL DELSPRITE(ALL):: CALL CLEAR :: CALL SCREEN(15)
390 CALL CHAR(88,"FFFFFFFFFFFFFFFF")
400 CALL CHAR(33,"FFFFFFFFFFFFFFFF")
410 CALL CHAR(34,"0107070F3F3F7FFF")
420 CALL CHAR(35,"E0E0E0E0FCFCFEFF")
430 CALL CHAR(36,"FFFF7F3F0F0F0F03")
440 CALL CHAR(37,"FFFFFEFCF8F8E0C0")
450 CALL CHAR(40,"F35DF37DFFB7FFD5")
460 CALL CHAR(41,"0107070A35376BB7")
470 CALL CHAR(42,"E060E070B4EC7ED7")
480 CALL CHAR(43,"BDEE5D370E0B0F03")
490 CALL CHAR(44,"EFFB96AC78F8A0C0")
500 CALL CHAR(116,"181C7CE3E7F53B1C1F1F3F7C3B1C3C7C7070789CDC5CB870E0E0F0F870E0F
0F8")! AFFE
510 CALL CHAR(120,"0000000000000787863676F7F7E0C06000000000000000090A0BAFE9C08000
000")! DONALD ERLEDIGT
520 CALL CHAR(124,"000001030B070103170B0100000100030000B0C0C0502080C0E8F8F0A0209
0B8")! DAISY GERETTET
530 CALL CHAR(128,"FF5051535B575153575B5150505150FFFF0A8ACACA5A2A8ACAEAFAFAAA2A9
AFF")! DAISY GEFANGEN
540 CALL CHAR(132,"000000000103070F0F0F07070303010000000000080C0E0F0F0F0E0E0C0C0B
000")! KOKUSNUSS
550 CALL CHAR(136,"0103010101020405020000000000000000B000000000850A000000000000
000")! SCHLANGE FRIEDLICH
560 CALL CHAR(140,"0000000000000001020404040300000000000000000000850A000000000C0C
000")! SCHLANGE GREIFT AN
570 CALL COLOR(8,1,1):: CALL COLOR(1,1,1):: CALL COLOR(2,1,1)
580 CALL HCHAR(12,1,88,32*11):: CALL HCHAR(23,1,33,32*2)
590 CALL HCHAR(19,4,41):: CALL HCHAR(19,5,40,4):: CALL HCHAR(18,5,40,3):: CALL H
CHAR(18,8,42):: CALL HCHAR(17,5,40,3)
600 CALL HCHAR(16,5,41):: CALL HCHAR(16,6,40,2):: CALL HCHAR(16,8,44):: CALL HCH
AR(15,6,40,3):: CALL HCHAR(14,6,40,3)
610 CALL HCHAR(13,6,41):: CALL HCHAR(13,7,40,2):: CALL HCHAR(12,7,40,2):: CALL H
CHAR(11,7,41):: CALL HCHAR(11,8,40)
620 CALL HCHAR(11,9,44):: CALL HCHAR(10,7,43):: CALL HCHAR(10,8,40):: CALL HCHAR
(10,9,42):: CALL HCHAR(9,7,40)
630 CALL HCHAR(9,8,42):: CALL HCHAR(8,7,40):: CALL HCHAR(7,7,40)
640 CALL HCHAR(9,10,40,11):: CALL HCHAR(9,9,41):: CALL HCHAR(10,9,40):: CALL HCH
AR(10,10,44):: CALL HCHAR(9,21,44)
650 CALL HCHAR(8,12,41):: CALL HCHAR(8,13,40):: CALL HCHAR(8,14,42):: CALL HCHAR
(7,13,40)
```

DER GORILLA!

```
660 CALL HCHAR(8,17,41):: CALL HCHAR(8,18,40):: CALL HCHAR(8,19,42):: CALL HCHAR
(7,18,41):: CALL HCHAR(7,19,44)
670 CALL HCHAR(6,19,40)
680 CALL HCHAR(1,3,34):: CALL HCHAR(1,4,33,3):: CALL HCHAR(1,7,35):: CALL HCHAR(
2,2,34):: CALL HCHAR(2,3,33,5)
690 CALL HCHAR(2,8,35):: CALL HCHAR(3,2,36):: CALL HCHAR(3,3,33,5):: CALL HCHAR(
3,8,37):: CALL HCHAR(4,3,36)
700 CALL HCHAR(4,4,33,3):: CALL HCHAR(4,7,37):: CALL HCHAR(1,12,33,9):: CALL HCH
AR(1,11,34):: CALL HCHAR(1,21,35)
710 CALL HCHAR(2,11,33,11):: CALL HCHAR(2,10,34):: CALL HCHAR(2,22,35):: CALL HC
HAR(3,10,33,12):: CALL HCHAR(3,9,34):: CALL HCHAR(3,22,37)
720 CALL HCHAR(4,9,33,5):: CALL HCHAR(4,8,34):: CALL HCHAR(4,14,37):: CALL HCHAR
(5,8,36):: CALL HCHAR(5,9,33,5)
730 CALL HCHAR(5,14,35):: CALL HCHAR(6,9,36):: CALL HCHAR(6,10,33,4):: CALL HCHA
R(6,14,37):: CALL HCHAR(5,16,33,4)
740 CALL HCHAR(5,15,36):: CALL HCHAR(5,20,37):: CALL HCHAR(4,15,34):: CALL HCHAR
(4,16,33,5):: CALL HCHAR(4,21,37)
750 CALL HCHAR(6,5,36):: CALL HCHAR(6,6,33,2):: CALL HCHAR(6,8,35):: CALL HCHAR(
5,4,36):: CALL HCHAR(5,5,33,2)
760 CALL HCHAR(5,7,35):: CALL HCHAR(7,8,37)
770 CALL HCHAR(19,26,40,4):: CALL HCHAR(19,30,37):: CALL HCHAR(18,26,34):: CALL
HCHAR(18,27,33,6):: CALL HCHAR(17,27,34)
780 CALL HCHAR(17,28,33,5):: CALL HCHAR(16,28,34):: CALL HCHAR(16,29,33,2):: CAL
L HCHAR(16,31,35):: CALL HCHAR(15,30,34)
790 CALL HCHAR(15,31,35):: CALL HCHAR(20,29,43)
800 CALL COLOR(8,12,1):: CALL COLOR(1,3,1):: CALL COLOR(2,7,1)
810 DISPLAY AT(8,1)SIZE(4):"D";D
820 DISPLAY AT(2,22)SIZE(8):"SCORE" :: DISPLAY AT(3,22)SIZE(6):"";A :: DISPLAY A
T(5,22)SIZE(8):"TOP"
830 DISPLAY AT(6,22)SIZE(6):"";T :: DISPLAY AT(8,22)SIZE(8):"TIME" :: DISPLAY AT
(9,22)SIZE(3):"";TI
840 CALL SPRITE(#28,128,5,160,240)
850 CALL SPRITE(#27,136,2,150,220)
860 CALL SPRITE(#20,132,3,1,158,20+GZ,0)
870 CALL SPRITE(#19,132,3,1,42,18+GZ,0)
880 CALL SPRITE(#18,132,3,1,60,28+GZ,0)
890 CALL SPRITE(#17,132,3,1,78,19+GZ,0)
900 CALL SPRITE(#16,132,3,1,100,15+GZ,0)
910 CALL SPRITE(#15,132,3,1,123,20+GZ,0)
920 CALL SPRITE(#14,132,3,1,140,23+GZ,0)
930 CALL SPRITE(#13,116,2,50,112)
940 CALL SPRITE(#12,116,2,106,222)
950 CALL SPRITE(#1,104,5,160,8)
960 CALL SOUND(500,196,5,262,5,330,5):: CALL SOUND(1000,391,5,523,5,659,5)
970 TI=TI-1 :: DISPLAY AT(9,22)SIZE(3):"";TI :: IF TI=0 THEN 1120
980 CALL JOYST(1,X,Y):: IF X=4 THEN 1030 ELSE IF X=-4 THEN 1040
990 CALL MOTION(#1,0,0)
1000 CALL COINC(ALL,C):: IF C THEN 1120
1010 XS=RND*1 :: IF XS>.9 THEN 1070
1020 GOTO 970
1030 CALL MOTION(#1,0,5):: CALL PATTERN(#1,104):: CALL COINC(ALL,C):: IF C THEN
1120 :: CALL PATTERN(#1,108):: GOTO 1000
1040 CALL MOTION(#1,0,-5):: CALL PATTERN(#1,96):: CALL COINC(ALL,C):: IF C THEN
1120 :: CALL PATTERN(#1,100)
1050 CALL POSITION(#1,A1,B1):: IF B1<16 THEN 1060 ELSE 1000
1060 CALL LOCATE(#1,160,24):: GOTO 1000
1070 CALL PATTERN(#27,140)
1080 FOR I=1 TO 5
1090 CALL COINC(#1,#27,12,C):: IF C THEN 1120
1100 NEXT I
1110 CALL PATTERN(#27,136):: GOTO 970
1120 CALL COINC(#1,#28,16,C):: IF C THEN 1180
1130 CALL MOTION(#1,0,0):: CALL PATTERN(#1,120)
1140 CALL SOUND(1000,260,5,296,5,330,5)
1150 FOR I=1 TO 300 :: NEXT I
1160 D=D-1 :: DISPLAY AT(8,2)SIZE(3):"";D :: IF D=0 THEN 1220 :: CALL PATTERN(#1
,104):: CALL LOCATE(#1,160,16)
1170 TI=50 :: GOTO 970
1180 CALL MOTION(#1,0,0):: CALL PATTERN(#28,124)
1190 CALL SOUND(500,196,5,262,5,330,5):: CALL SOUND(500,391,5,523,5,659,5):: CAL
L SOUND(500,783,5,1046,5,1567,5)
1200 FOR I=1 TO 300 :: NEXT I
1210 A=A+100+TI :: DISPLAY AT(3,22)SIZE(6):"";A :: GZ=GZ+1 :: TI=50 :: GOTO 840
1220 DISPLAY AT(11,12)SIZE(9):"GAME OVER"
1230 CALL SOUND(250,880,0):: CALL SOUND(250,783,0):: CALL SOUND(125,659,0):: CAL
L SOUND(175,523,0):: CALL SOUND(125,659,0)
1240 CALL SOUND(250,587,0):: CALL SOUND(125,659,0):: CALL SOUND(125,587,0):: CAL
L SOUND(125,587,0):: CALL SOUND(125,783,0)
1250 CALL SOUND(125,659,0):: CALL SOUND(125,587,0):: CALL SOUND(125,587,0):: CAL
L SOUND(125,523,0):: CALL SOUND(125,659,0)
1260 CALL SOUND(125,587,0):: CALL SOUND(125,587,0):: CALL SOUND(125,783,0):: CAL
L SOUND(125,659,0):: CALL SOUND(125,587,0)
1270 CALL SOUND(125,587,0):: CALL SOUND(250,880,0):: CALL SOUND(250,783,0):: CAL
L SOUND(125,659,0):: CALL SOUND(175,523,0)
1280 CALL SOUND(125,659,0):: CALL SOUND(500,587,0)
1290 IF A>T THEN T=A :: DISPLAY AT(6,22)SIZE(6):"";T
1300 CALL KEY(0,K,S):: IF S=0 THEN 1300
1310 CALL DELSPRITE(ALL):: CALL COLOR(8,2,1):: GOTO 110
```



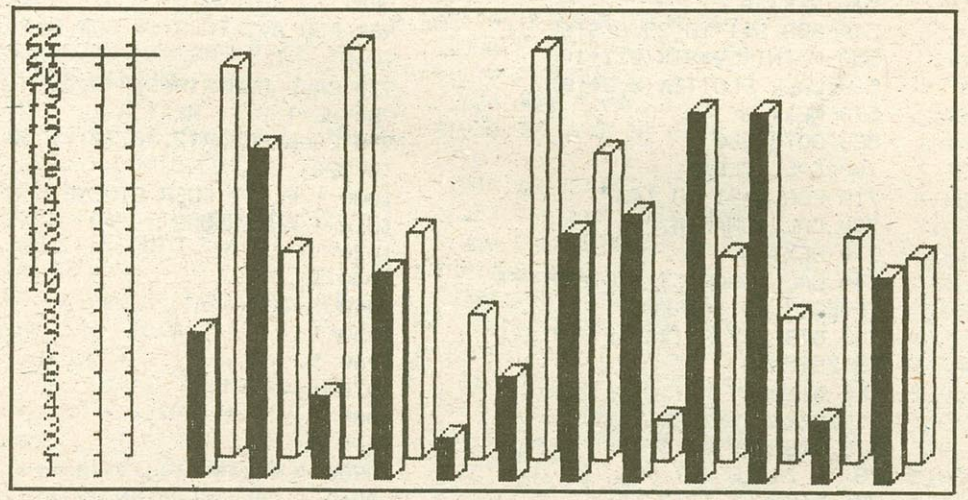


Hardcopy von Säulendiagrammen

für den TI-99 mit Extended Basic

**Das folgende kurze Programm ermöglicht es, Hardcopy's von Säulendiagrammen auf einem Seikosha-GP100A auszudrucken.
Die Säulen werden in Blockgrafik dargestellt, es können dabei 2 Werte verglichen werden.**

Zu beachten:
**Der Druckerbefehl in Zeile 280 gilt für das Interface der Fa. Klein Computer, doch dürfte der Betrieb mit anderen Interfaces ebenfalls gewährleistet sein.
Werte, die in den Zeilen 240 und 260 festgelegten Bildschirmbereich überschreiten, müssen entsprechend umgerechnet werden. In den Zeilen 130-190 werden die Zeichen definiert, in den Zeilen 200, 380-420 und 460 dem ASCII-Code zugeordnet. Der Zufallsgenerator in Zeile 220 dient nur der Demonstration, durch Änderung der Schrittweite in dieser Zeile kann der Säulenabstand verändert werden. Nach Löschen des Zufallsgenerators und der Bildschirmanpassung kann das Programm als Unterprogramm verwendet werden.**



```
120 CALL CLEAR :: CALL SCREEN(2):: FOR I=2 TO 12 :: CALL COLOR(I,13,2):: NEXT I
:: CALL COLOR(0,13,2,1,16,2,13,16,2,14,14,2)
130 DATA "F1F1F1F1F1F1F1F1","000000001F234589","F1F1F1F1F2F4F8","404040404040400"
140 DATA 2,192,1,255,7,128
150 DATA 1,192,1,224,1,208,1,200,1,196,1,196,1,164,1,148,1,140,1,252
160 DATA 6,255,3,128,1,255
170 DATA 1,255,4,128,1,255,3,128,1,255
180 DATA 6,255,1,160,1,144,1,136,1,143
190 DATA 1,255,4,192,1,255,1,160,1,144,1,136,1,143
200 FOR I=128 TO 130 :: READ A$ :: CALL CHAR(I,A$):: CALL CHAR(I+8,A$):: NEXT I
:: READ A$ :: CALL CHAR(33,A$)
210 CALL VCHAR(2,3,33,21)::CALL VCHAR(1,4,33,21)
220 FOR X=6 TO 28 STEP 2 :: RANDOMIZE :: Y1=INT(20*RND)+1 :: Y2=INT(20*RND)+1
230 IF Y1=0 THEN GOTO 250
240 CALL VCHAR(22-Y1,X,129):: CALL VCHAR(23-Y1,X,128,Y1)
:: CALL VCHAR(22,X,130)
250 IF Y2=0 THEN GOTO 270
260 CALL VCHAR(21-Y2,X+1,137)::CALL VCHAR(22-Y2,X+1,136,Y2):: CALL VCHAR(21,X+1,138)
270 NEXT X
280 OPEN #1:"RS232.DA=8.PA=N.CR.LF"
290 FOR Y=1 TO 22 :: PRINT #1:CHR$(27);CHR$(16);CHR$(0);CHR$(60):: PRINT #1,USING "##":23-Y;:: PRINT #1:CHR$(8);
300 P=P+8 :: FOR X=2 TO 29 :: CALL GCHAR(Y,X,Q)
310 IF Q=32 THEN GOTO 430 ELSE IF Q=33 THEN GOTO 460
320 ON Q-127 GOTO 390,380,400,32767,32767,32767,32767,32767,410,380,420
330 NEXT X
340 PRINT #1:CHR$(28);CHR$(480-P);CHR$(128)
350 L=P :: P=0 :: PRINT #1:CHR$(15):: NEXT Y
360 PRINT #1:CHR$(15)
370 CLOSE #1 :: GOTO 490
380 RESTORE 150 :: FOR N=1 TO 10 :: READ A,Z :: GOSUB 470 :: NEXT N :: GOTO 330
390 RESTORE 160 :: FOR N=1 TO 3 :: READ A,Z :: GOSUB 470 :: NEXT N :: GOTO 330
400 RESTORE 180 :: FOR N=1 TO 5 :: READ A,Z :: GOSUB 470 :: NEXT N :: GOTO 330
410 RESTORE 170 :: FOR N=1 TO 5 :: READ A,Z :: GOSUB 470 :: NEXT N :: GOTO 330
420 RESTORE 190 :: FOR N=1 TO 7 :: READ A,Z :: GOSUB 470 :: NEXT N :: GOTO 330
430 PRINT #1:CHR$(28);CHR$(10);CHR$(128);
440 P=P+10
450 GOTO 330
460 RESTORE 140 :: FOR N=1 TO 3 :: READ A,Z :: GOSUB 470 :: NEXT N :: GOTO 330
470 P=P+A :: PRINT #1:CHR$(28);CHR$(A);CHR$(Z);
480 RETURN
490 END
```

Punkte plotten

für den TI-99/4A mit Extended Basic

Mit diesem Programm kann jeder Punkt der 256 x 192 Auflösung angesteuert werden. Der Befehl "Call plotten" (Cy-Achse, X-Achse) findet dazu Verwendung. Die Zeilen 100 bis 1000 zeigen einige Anwendungsbeispiele dieses Befehls. So wird per Joystick der "Pinsel" über den Bildschirm bewegt, gleichzeitig werden Linien gezogen. Ein Druck auf die Feuertaste bewirkt, daß zwar keine Linien gezogen werden, der "Pinsel" sich aber dennoch bewegen läßt. Die Zeilen 1000 bis 28000 lassen für einige Ideen Anwendung finden. (Statistik, Grafik usw.)

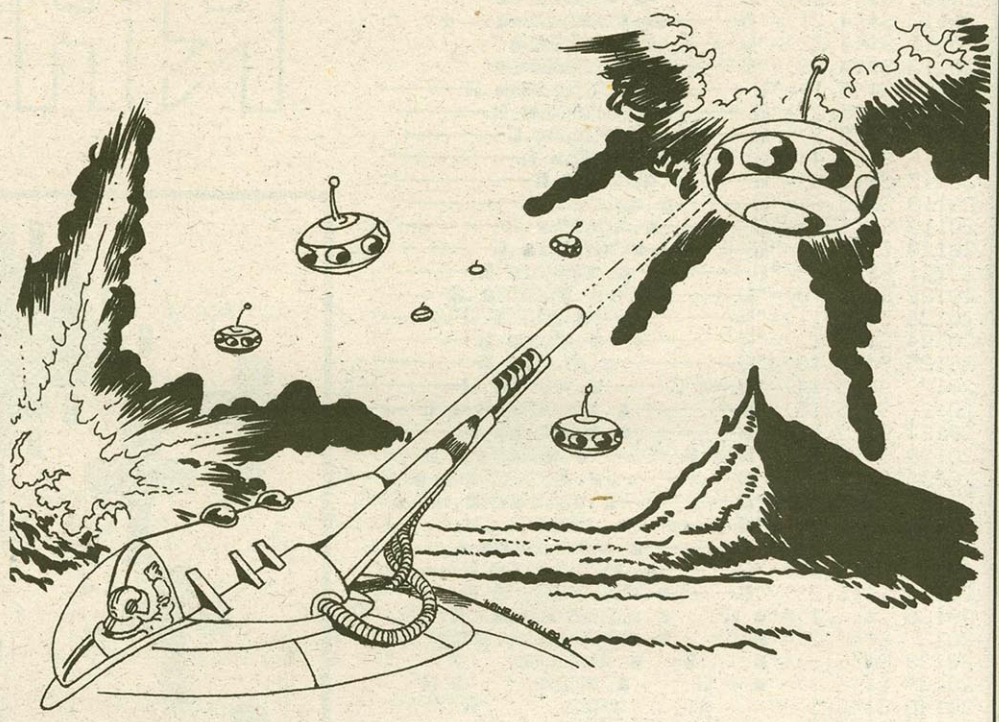
Zu bemerken:
Lediglich 110 Zeichen lassen sich mit diesem Programm neu definieren.

```
100 CALL CLEAR
110 FOR A=0 TO 14
120 CALL COLOR(A,2,8)
130 NEXT A
140 CALL SCREEN(16)
150 CALL CHAR(129,"00FFFF",4
3,"3C4299A1A199423C")
160 DISPLAY AT(2,4)BEEP:"192
*256 PUNKTE PLOTTEN"
170 DISPLAY AT(3,4):""
180 DISPLAY AT(4,6):"+ 31.JU
LI 1983,VON"
190 DISPLAY AT(5,6):"MICHAEL
 VORWIEGER"
200 DISPLAY AT(6,6):"CHATTEN
STRASSE 6."
210 DISPLAY AT(7,6):"6238 HO
FHEIM Ts."
220 DISPLAY AT(10,1):""
230 DISPLAY AT(11,4):"DRUEKE
"
240 DISPLAY AT(13,4):"1 FUER
 MALEN"
250 DISPLAY AT(15,4):"2 FUER
 SINUSKURVE"
260 DISPLAY AT(17,4):"3 FUER
 ZEICHEN DEFINIEREN"
270 DISPLAY AT(19,4)BEEP:"4
FUER EIGENE ANWENDUNG"
280 CALL KEY(0,K,S):: IF S=0
 OR K<49 OR K>52 THEN 280
290 ON K-48 GOTO 300,500,700
,1000
300 CALL CLEAR
310 FOR A=34 TO 143
320 CALL CHAR(A,"")
330 NEXT A
340 CALL CHAR(33,"8040303810
")
350 W=98 :: S=128
360 CALL SPRITE(#1,33,7,W,S)
370 CALL JOYST(2,X,Y):: CALL
 KEY(2,KK,SS):: IF SS+X+Y=0
THEN 370
380 IF X=4 THEN S=S+1
390 IF X=-4 THEN S=S-1
400 IF Y=4 THEN W=W-1
410 IF Y=-4 THEN W=W+1
420 CALL LOCATE(#1,W,S)
430 IF KK=18 THEN 370
440 CALL PLOTTEN(W,S)
450 GOTO 370
500 CALL CLEAR
510 FOR A=34 TO 143
520 CALL CHAR(A,"")
530 NEXT A
540 FOR S=1 TO 25.6 STEP .1
550 W=INT(50*SIN(S))+100
560 CALL PLOTTEN(W,S*10)
570 NEXT S
580 GOTO 580
700 CALL CLEAR
710 FOR A=91 TO 143
720 CALL CHAR(A,"")
730 NEXT A
740 CALL CHAR(33,"3C7EFFFFFF
FF7E3C",37,"00FFFF")
750 DISPLAY AT(2,1):"AKTUELL
ES ZEICHEN   :"
760 DISPLAY AT(3,1):"%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%"
770 DISPLAY AT(5,1):" 123456
78    ZEILE  :"
780 DISPLAY AT(7,1):"
      SPALTE :"
790 FOR A=1 TO 8
800 CALL HCHAR(5+A,3,A+48)
810 NEXT A
820 ACCEPT AT(2,23)SIZE(3)VA
LIDATE(DIGIT)BEEP:ZEICHEN ::
 CALL HCHAR(2,30,ZEICHEN
)
830 IF ZEICHEN>143 OR ZEICHE
N<33 THEN 820
840 FOR Z=1 TO 64
850 ACCEPT AT(5,23)SIZE(1)VA
LIDATE(DIGIT)BEEP:ZEILE
860 IF ZEILE>8 OR ZEILE<1 TH
EN 850
870 ACCEPT AT(7,23)SIZE(1)VA
LIDATE(DIGIT)BEEP:SPALTE
880 IF SPALTE=0 THEN 930
890 IF SPALTE>8 THEN 870
900 CALL PLOT(ZEILE,SPALTE,Z
EICHEN)
910 CALL HCHAR(ZEILE+5,SPALT
E+3,33)
920 NEXT Z
930 X=X+1
940 IF X<30 THEN 950 :: Y=Y+
1 :: X=1
950 CALL HCHAR(Y+20,X+2,ZEIC
HEN)
960 FOR A=1 TO 8 :: FOR B=1
TO 8
970 CALL HCHAR(A+5,B+3,32)
980 NEXT B :: NEXT A
990 CALL HCHAR(2,30,32):: GO
TO 820
1000 ! PLATZ FUER EIGENE
1010 ! ANWENDUNG
1020 !
1030 !
1040 !
1050 !
1060 !
1070 !
1080 !
1090 !
28900 ! ******************
28910 ! CALL PLOTTEN (Y,X)
28920 ! ROTINE / ANFANG
28930 ! 4 K.BYTE PROGRAMM
28940 ! ******************
28950 !
29000 SUB PLOTTEN(WA,SE)
29010 IF WA>192 OR WA<1 OR S
E>256 OR SE<1 THEN 29230
29020 WGC=25
29030 SGC=33
29040 FOR WC=184 TO 0 STEP -
8
29050 WGC=WGC-1
29060 IF WA>WC THEN 29080
29070 NEXT WC
29080 FOR SC=248 TO 0 STEP -
8
29090 SGC=SGC-1
29100 IF SE>SC THEN 29120
29110 NEXT SC
```

```
29120 CALL GCHAR(WGC,SGC,TGC
)
29130 IF TGC=32 THEN GOSUB 2
9180
29140 TW=WA-WC
29150 TS=SE-SC
29160 CALL PLOT(TW,TS,TGC)
29170 GOTO 29240
29180 IF Z+33=143 THEN 29230
29190 Z=Z+1
29200 CALL HCHAR(WGC,SGC,Z+3
3)
29210 TGC=Z+33
29220 RETURN
29230 CALL SOUND(200,220,0)
29240 SUBEND
30000 SUB PLOT(W,S,Z)
30010 IF S>4 THEN 30040
30020 WA=W*2-1 :: SE=S :: A=
1
30030 GOTO 30050
30040 WA=W*2 :: SE=S-4
30050 !
30060 CALL CHARPAT(Z,A$)
30070 B$=SEG$(A$,WA,1)
30080 IF B$="F" THEN 30410
30090 IF B$="0" THEN CALL Z0
(WA,SE,Z):: GOTO 30410
30100 !
30110 IF B$="1" THEN CALL Z1
(WA,SE,Z):: GOTO 30410
30120 !
30130 IF B$="2" THEN CALL Z2
(WA,SE,Z):: GOTO 30410
30140 !
30150 IF B$="3" THEN CALL Z3
(WA,SE,Z):: GOTO 30410
30160 !
30170 IF B$="4" THEN CALL Z4
(WA,SE,Z):: GOTO 30410
30180 !
30190 IF B$="5" THEN CALL Z5
(WA,SE,Z):: GOTO 30410
30200 !
30210 IF B$="6" THEN CALL Z6
(WA,SE,Z):: GOTO 30410
30220 !
30230 IF B$="7" THEN CALL Z7
(WA,SE,Z):: GOTO 30410
30240 !
30250 IF B$="8" THEN CALL Z8
(WA,SE,Z):: GOTO 30410
30260 !
30270 IF B$="9" THEN CALL Z9
(WA,SE,Z):: GOTO 30410
30280 !
30290 IF B$="A" THEN CALL ZA
(WA,SE,Z):: GOTO 30410
30300 !
30310 IF B$="B" THEN CALL ZB
(WA,SE,Z):: GOTO 30410
30320 !
30330 IF B$="C" THEN CALL ZC
(WA,SE,Z):: GOTO 30410
30340 !
30350 IF B$="D" THEN CALL ZD
(WA,SE,Z):: GOTO 30410
30360 !
30370 IF B$="E" THEN CALL ZE
(WA,SE,Z):: GOTO 30410
30380 !
30410 CALL SOUND(200,770,0)
30420 !
30430 SUBEND
31000 SUB Z0(WA,SE,Z)
31010 !
31020 IF SE=1 THEN C$="8"
31030 IF SE=2 THEN C$="4"
31040 IF SE=3 THEN C$="2"
31050 IF SE=4 THEN C$="1"
31060 CALL C(WA,SE,Z,C$)
31070 SUBEND
31080 SUB Z1(WA,SE,Z)
31090 IF SE=4 THEN 31140
31100 IF SE=3 THEN C$="3"
31110 IF SE=2 THEN C$="5"
31120 IF SE=1 THEN C$="9"
31130 CALL C(WA,SE,Z,C$)
31140 SUBEND
31150 SUB Z2(WA,SE,Z)
31160 IF SE=3 THEN 31210
31170 IF SE=1 THEN C$="A"
31180 IF SE=2 THEN C$="6"
31190 IF SE=4 THEN C$="3"
31200 CALL C(WA,SE,Z,C$)
31210 SUBEND
31220 SUB Z3(WA,SE,Z)
31230 IF SE=3 OR SE=4 THEN 3
1270
31240 IF SE=1 THEN C$="B"
31250 IF SE=2 THEN C$="7"
31260 CALL C(WA,SE,Z,C$)
31270 SUBEND
31280 SUB Z4(WA,SE,Z)
31290 IF SE=2 THEN 31340
31300 IF SE=1 THEN C$="C"
31310 IF SE=3 THEN C$="6"
31320 IF SE=4 THEN C$="5"
31330 CALL C(WA,SE,Z,C$)
31340 SUBEND
31350 SUB Z5(WA,SE,Z)
31360 IF SE=2 OR SE=4 THEN 3
1400
31370 IF SE=1 THEN C$="D"
31380 IF SE=3 THEN C$="7"
31390 CALL C(WA,SE,Z,C$)
31400 SUBEND
31410 SUB Z6(WA,SE,Z)
31420 IF SE=2 OR SE=3 THEN 3
1460
31430 IF SE=1 THEN C$="E"
31440 IF SE=4 THEN C$="7"
31450 CALL C(WA,SE,Z,C$)
31460 SUBEND
31470 SUB Z7(WA,SE,Z)
31480 IF SE=2 OR SE=3 OR SE=
4 THEN 31510
31490 IF SE=1 THEN C$="F"
31500 CALL C(WA,SE,Z,C$)
31510 SUBEND
31520 SUB Z8(WA,SE,Z)
31530 IF SE=1 THEN 31580
31540 IF SE=2 THEN C$="C"
31550 IF SE=3 THEN C$="A"
31560 IF SE=4 THEN C$="9"
31570 CALL C(WA,SE,Z,C$)
31580 SUBEND
31590 SUB Z9(WA,SE,Z)
31600 IF SE=1 OR SE=4 THEN 3
1640
31610 IF SE=2 THEN C$="D"
31620 IF SE=3 THEN C$="B"
31630 CALL C(WA,SE,Z,C$)
31640 SUBEND
31650 SUB ZA(WA,SE,Z)
31660 IF SE=1 OR SE=3 THEN 3
1700
31670 IF SE=2 THEN C$="E"
31680 IF SE=4 THEN C$="B"
31690 CALL C(WA,SE,Z,C$)
31700 SUBEND
31710 SUB ZB(WA,SE,Z)
31720 IF SE=1 OR SE=3 OR SE=
4 THEN 31750
31730 IF SE=2 THEN C$="F"
31740 CALL C(WA,SE,Z,C$)
31750 SUBEND
31760 SUB ZC(WA,SE,Z)
31770 IF SE=1 OR SE=2 THEN 3
1810
31780 IF SE=3 THEN C$="E"
31790 IF SE=4 THEN C$="D"
31800 CALL C(WA,SE,Z,C$)
31810 SUBEND
31820 SUB ZD(WA,SE,Z)
31830 IF SE=1 OR SE=2 OR SE=
4 THEN 31860
31840 IF SE=3 THEN C$="F"
31850 CALL C(WA,SE,Z,C$)
31860 SUBEND
31870 SUB ZE(WA,SE,Z)
31880 IF SE=4 THEN 31890 ELS
E 31910
31890 C$="F"
31900 CALL C(WA,SE,Z,C$)
31910 SUBEND
32000 SUB C(WA,SE,Z,C$)
32010 CALL CHARPAT(Z,A$)
32020 B$=SEG$(A$,1,WA-1)&C$&
SEG$(A$,WA+1,16-WA)
32030 CALL CHAR(Z,B$)
32040 SUBEND
32050 !          60/
32060 ! ENDE      0
32070 !          / 6
```

Planetenverteidigung

für den TI-99/4A



Das Spiel besteht aus zwei Bildern. Im ersten Bild muß man die feindlichen Gegner ins Visier bringen und abschießen. Im zweiten Bild muß das eigene blaue Raumschiff sicher auf der Plattform gelandet werden. Es stehen 3000 Energieeinheiten zur Verfügung. Jeder Schuß kostet 100 Energieeinheiten. Wenn die Bildschirmfarbe wechselt, so mußten Sie einen gegnerischen Treffer einstecken, wofür Ihnen ebenfalls 100 Energiepunkte abgezogen werden. In der ersten von insgesamt vier Runden, müssen 8 Gegner beseitigt werden.
Haben Sie das erste Bild erfolgreich geschafft, so müssen Sie nun auf der Planetenoberfläche landen. Hier fliegen Sie nun durch die Anziehungskraft automatisch nach unten. Um nun durch den Gang hindurch zur Landeplattform zu gelangen, muß ein Gegenschub erzeugt werden. Dafür werden jedesmal 25 Energiepunkte abgezogen. Nach erfolgreicher Landung beginnt das erste Bild von neuem zu starten, jedoch mit 10 Gegner und 250 Energiepunkten mehr. Die Anzahl der Gegner steigert sich bis zu 14 bei 3750 Energieeinheiten.

Steuerung:
Im ersten Bild wird mit Joystick 1 in alle Richtungen, auch diagonal, gesteuert. Der Feuerknopf dient dem Abschießen der Gegner.
Im zweiten Bild kann nur nach links und rechts gesteuert werden. In Verbindung mit dem "nach unten schweben" ergibt sich so immer eine Diagonalsteuerung. Der Feuerknopf dient hier der Aktivierung des Gegenschubs.
Wem das Spiel zu leicht oder zu schwer ist, kann die Werte in folgenden Zeilen ändern.
290 u = 6 (steht für Anfangszahl der Gegner)
300 EM=DM+250 (steht für den Energiezuwachs je Runde)
310U=U+2 (steht für den Zuwachs der Gegner je Runde)
320 EN=2750+EM (Steht für die Anfangsenergie in der ersten Runde)

```
100 CALL CLEAR
110 CALL SCREEN(14)
120 FOR Z=1 TO 16
130 CALL COLOR(Z,16,1)
140 NEXT Z
150 PRINT "      ***********
*****":"      *
 *":"      * Planetenver
- *":"      *  teidigung   *
":
160 PRINT "      *
   *":"      **************
**":::::::::::::"Juni 1
984 by Klaus Hofmann"
170 FOR Z=1 TO 750
180 NEXT Z
190 CALL CLEAR
200 PRINT "Willkommen bei Pl
anetenver- teidigung.":"Ges
pielt wird mit Joystick
1"::"Ziel ist es, die gruene
n "
210 PRINT "Gegner in Ihr vis
ier zu     bringen! Wenn Sie
 die erste Angriffswelle
 ueberstanden  haben, muesse
n";
220 PRINT " Sie ver-":"suche
n auf der Landeplatt-  form
aufzusetzen.":"Uebrigen
s!!!"::"Sie koennen das Spie
l ";
230 PRINT "nicht unterbreche
n!"::"Fertig dann ""F I R E"
""
240 CALL KEY(1,K,S)
250 IF (S=0)+(K<>18)THEN 240
260 FOR Z=1 TO 16
270 CALL COLOR(Z,1,1)
280 NEXT Z
290 U=6
300 EM=EM+250
310 U=U+2
320 EN=2750+EM
330 CALL CLEAR
340 CALL SCREEN(2)
350 DATA FFFFFF7F7F7F7F3F,3F
3F3F3F1F1F1F0F,0F07070303030
101,FF7F7F3F3F1F0F07,070
3010100,FFFFFFFFFF7F3F1F
360 DATA 0F07030100,FFFFFFFF
FF7F3F1F,0F030100
370 DATA FFFFFF7F3F1F1F0F,07
0300,FFFFFF7F3F1F0703,7F1F07
0100,FFFFFFFF7F1F0703,FF
3F070100
380 DATA FFFFFFFF7F3F0F03,FF
3F0F0300,FFFFFFFF7F0700,FFFF
FFFFFFFF7F07
390 RESTORE 350
400 FOR Z=104 TO 122
410 READ A$
420 CALL CHAR(Z,A$)
430 NEXT Z
440 CALL CHAR(41,"4000000000
000001")
450 CALL CHAR(42,"0000000010
")
460 CALL CHAR(43,"040")
470 DATA FFFFDFDFEFF7F9FF,FF
FFFDFBFB8DFFFF,FFFFC3BFFFFFF
FFF,FFCFB7FBFBFDFDFF,EFE
FEFDFFFFFFFFF,FDFBFFFFFFC7BF
7F
480 DATA FFFFFFFFFFFFFFFF
490 RESTORE 470
500 FOR Z=152 TO 158
510 READ A$
520 CALL CHAR(Z,A$)
530 NEXT Z
540 DATA 03060C183060C080,C0
6030180C060301,000000183C66C
381,A22845B81AA80C42,181
81818243C66C3,00182442FFFF24
```

```
42
550 DATA 428199A5E7998142,C3
661824241866C3,003C7EA57E3C0
0
560 RESTORE 540
570 FOR Z=132 TO 140
580 READ A$
590 CALL CHAR(Z,A$)
600 NEXT Z
610 CALL CHAR(144,"000000FFF
F00")
620 CALL CHAR(145,"181818181
8181818")
630 CALL CHAR(96,"FFFFFFFFFF
FFFFFF")
640 CALL HCHAR(1,1,96,768)
650 FOR Z=4 TO 21
660 CALL HCHAR(Z,5,32,24)
670 NEXT Z
680 FOR Z=1 TO 20
690 RANDOMIZE
700 O=INT((28-5+1)*RND)+5
710 P=INT((21-4+1)*RND)+4
720 Q=INT((43-41+1)*RND)+41
730 CALL HCHAR(P,O,Q)
740 NEXT Z
750 CALL HCHAR(12,11,144,4)
760 CALL HCHAR(12,18,144,4)
770 CALL VCHAR(8,16,145,3)
780 CALL VCHAR(14,16,145,3)
790 Z1=103
800 DATA 4,16,5,16,6,16,7,17
,8,17,8,18,9,18,9,19,10,19,1
0,20,11,20,11,21,12,22,1
2,23,13,24,13,25
810 DATA 14,26,14,27,14,28
820 RESTORE 800
830 FOR Z=1 TO 19
840 Z1=Z1+1
850 READ A,B
860 CALL HCHAR(A,B,Z1)
870 NEXT Z
880 FOR Z=4 TO 6
890 CALL HCHAR(Z,17,158,12)
900 NEXT Z
910 DATA 7,18,11,8,19,10,9,2
0,9,10,21,8,11,22,7,12,24,5,
13,26,3
920 RESTORE 910
930 FOR Z=1 TO 7
940 READ A,B,C
950 CALL HCHAR(A,B,158,C)
960 NEXT Z
970 R=16
980 DATA 4,26,153,5,19,156,5
,21,152,6,26,157,8,19,153,8,
22,154,8,24,157
990 DATA 11,22,155,11,26,152
,12,28,156,13,26,155
1000 RESTORE 980
1010 FOR Z=1 TO 11
1020 READ A,B,C
1030 CALL HCHAR(A,B,C)
1040 NEXT Z
1050 A$="ENERGIE"
1060 CALL COLOR(9,5,1)
1070 FOR Z=5 TO 8
1080 CALL COLOR(Z,16,5)
1090 NEXT Z
1100 CALL COLOR(13,7,1)
1110 CALL COLOR(15,14,1)
1120 CALL COLOR(16,12,10)
1130 CALL COLOR(10,12,1)
1140 CALL COLOR(11,12,1)
1150 CALL COLOR(12,12,1)
1160 CALL COLOR(2,R,1)
1170 CALL COLOR(3,16,5)
1180 CALL COLOR(4,16,5)
1190 GOSUB 5250
1200 GOSUB 5190
1210 GOSUB 2390
1220 CALL HCHAR(V,H,F)
1230 D=0
1240 FOR Z=1 TO 4
1250 CALL SOUND(-450,-5,16)
1260 CALL JOYST(1,X,Y)
1270 CALL KEY(1,K,S)
1280 IF K=18 THEN 1890
1290 IF X=-4 THEN 1500
1300 IF X=4 THEN 1370
1310 IF Y=4 THEN 1760
1320 IF Y=-4 THEN 1630
1330 NEXT Z
1340 D=D+1
1350 IF D=E THEN 2250
1360 GOTO 1240
1370 IF Y=4 THEN 4520
1380 IF Y=-4 THEN 4760
1390 H=H+1
1400 CALL SOUND(-450,-5,16)
1410 IF (H<4)+(H>29)+(V<3)+(
V>22)THEN 1470
1420 CALL HCHAR(V,H-1,G)
1430 CALL GCHAR(V,H,G)
1440 GOSUB 2470
1450 IF G=96 THEN 1470
1460 CALL HCHAR(V,H,F)
1470 D=D+1
1480 IF D=E THEN 2250
1490 GOTO 1240
1500 IF Y=4 THEN 4640
1510 IF Y=-4 THEN 4880
1520 H=H-1
1530 CALL SOUND(-450,-5,16)
1540 IF (H<4)+(H>29)+(V<3)+(
V>22)THEN 1600
1550 CALL HCHAR(V,H+1,G)
1560 CALL GCHAR(V,H,G)
1570 GOSUB 2470
1580 IF G=96 THEN 1600
1590 CALL HCHAR(V,H,F)
1600 D=D+1
1610 IF D=E THEN 2250
1620 GOTO 1240
1630 IF X=4 THEN 4760
1640 IF X=-4 THEN 4880
1650 V=V+1
1660 CALL SOUND(-450,-5,16)
1670 IF (V<3)+(V>22)+(H<4)+(
H>29)THEN 1730
1680 CALL HCHAR(V-1,H,G)
1690 CALL GCHAR(V,H,G)
1700 GOSUB 2470
1710 IF G=96 THEN 1730
1720 CALL HCHAR(V,H,F)
1730 D=D+1
1740 IF D=E THEN 2250
1750 GOTO 1240
1760 IF X=4 THEN 4520
1770 IF X=-4 THEN 4640
1780 V=V-1
1790 CALL SOUND(-450,-5,16)
1800 IF (V<3)+(V>22)+(H<4)+(
H>29)THEN 1860
1810 CALL HCHAR(V+1,H,G)
1820 CALL GCHAR(V,H,G)
1830 GOSUB 2470
1840 IF G=96 THEN 1860
1850 CALL HCHAR(V,H,F)
1860 D=D+1
1870 IF D=E THEN 2250
1880 GOTO 1240
1890 CALL GCHAR(12,16,I)
1900 H1=6
1910 H2=26
1920 FOR Z=21 TO 13 STEP -1
1930 H1=H1+1
1940 H2=H2-1
1950 CALL HCHAR(Z,H1,132)
1960 CALL HCHAR(Z,H2,133)
1970 NEXT Z
1980 CALL HCHAR(12,16,134)
1990 CALL SOUND(100,-7,0)
2000 H1=6
2010 H2=26
2020 FOR Z=21 TO 13 STEP -1
2030 H1=H1+1
2040 H2=H2-1
2050 CALL HCHAR(Z,H1,32)
2060 CALL HCHAR(Z,H2,32)
2070 NEXT Z
2080 IF I<>32 THEN 2150
2090 CALL HCHAR(12,16,32)
2100 IF (H<4)+(H>29)+(V<3)+(
V>22)THEN 2120
2110 CALL HCHAR(V,H,F)
2120 EN=EN-100
2130 GOSUB 5190
2140 GOTO 1240
2150 CALL HCHAR(12,16,135)
2160 FOR L=0 TO 30
2170 CALL SOUND(-100,-6,L)
2180 NEXT L
2190 CALL HCHAR(12,16,32)
2200 EN=EN-100
2210 T=T+1
2220 GOSUB 5190
2230 IF T=U THEN 2530
2240 GOTO 1210
2250 D=0
2260 EN=EN-100
2270 FOR Z=3 TO 14
2280 CALL SCREEN(Z)
2290 CALL SOUND(-20,-3,0)
2300 NEXT Z
2310 CALL SCREEN(11)
2320 CALL SOUND(-20,-3,0)
2330 CALL SCREEN(2)
2340 CALL SOUND(-20,-3,0)
2350 RANDOMIZE
2360 E=INT((18-11+1)*RND)+11
2370 GOSUB 5190
2380 GOTO 1240
2390 RANDOMIZE
2400 H=INT((28-5+1)*RND)+5
2410 V=INT((21-4+1)*RND)+4
2420 F=INT((140-136+1)*RND)+
136
2430 CALL GCHAR(V,H,G)
2440 GOSUB 2470
2450 E=INT((18-11+1)*RND)+11
2460 RETURN
2470 IF G<152 THEN 2500
2480 N=12
2490 GOTO 2510
2500 N=1
2510 CALL COLOR(14,3,N)
2520 RETURN
2530 CALL CLEAR
2540 FOR Z=1 TO 16
2550 CALL COLOR(Z,1,1)
2560 NEXT Z
2570 DATA 002024347E7FFFFF,0
000000CD2C1E3FF,8891524C283A
FAFF,00100E02071FFFFF,10
08089090E0F8FF,905051393B7BF
FFF
2580 DATA 00000080442E2F1F,1
F3F3F7F7F7FFFFF,40E0E0F0F0F0
F0FA,FAF1F1F9FEFEFFFF,FF
FFFF7F1F1F0F03,FFFFFFFBE9A42
242
2590 DATA FFFF5E1199146,FFFF
0C10A8C8161,80C0C0E0F8FEFFFF
,000000000000C0F0,F0F0F0
F0F0F0F0F0,FFFFFEFEFCFCF0F
2600 DATA FFFFFCF0E0F0C08,F0
80,E0C0C0E0E0F3FFFF
2610 RESTORE 2570
2620 FOR Z=97 TO 117
2630 READ A$
2640 CALL CHAR(Z,A$)
2650 NEXT Z
2660 CALL CHAR(120,"FFC0A196
8C9C92A1")
2670 CALL CHAR(121,"A0C0C081
473F1F00")
2680 CALL CHAR(122,"F0688406
060E0E1E")
2690 CALL CHAR(123,"BE7CFCFC
FCF19C22")
2700 CALL CHAR(128,"FF603018
0C060301")
2710 CALL CHAR(129,"FF03060C
1830E0FF")
2720 CALL CHAR(130,"FFC06030
180C07FF")
2730 CALL CHAR(131,"FF060C18
3060C080")
2740 CALL CHAR(40,"0000070F1
F2262E2")
2750 CALL CHAR(41,"FFFF7F060
C1830F8")
2760 CALL CHAR(42,"0000E0F0F
8444647")
2770 CALL CHAR(43,"FFFFFE603
0180C1F")
2780 CALL HCHAR(11,1,96,21)
2790 CALL VCHAR(12,1,96,12)
2800 CALL HCHAR(24,1,96,22)
2810 CALL HCHAR(23,13,96,9)
2820 FOR Z=30 TO 32
2830 CALL VCHAR(12,Z,96,13)
2840 NEXT Z
2850 CALL HCHAR(17,7,96,12)
2860 CALL HCHAR(18,24,96,6)
2870 FOR Z=107 TO 110
2880 CALL HCHAR(12,Z-85,Z)
2890 CALL HCHAR(18,Z-100,Z)
2900 NEXT Z
2910 FOR Z=97 TO 102
2920 CALL HCHAR(16,Z-84,Z)
2930 NEXT Z
2940 CALL VCHAR(14,2,113,9)
2950 DATA 97,100,101,99,97,9
9
2960 RESTORE 2950
2970 FOR Z=1 TO 6
2980 READ Z1
2990 CALL HCHAR(16,Z+6,Z1)
3000 NEXT Z
3010 DATA 117,102,99,98,100,
101,97,102,98,97,104
3020 RESTORE 3010
3030 FOR Z=1 TO 11
3040 READ Z1
3050 CALL HCHAR(23,Z+1,Z1)
3060 NEXT Z
3070 DATA 103,105,100,101,98
,97,99,99,97,98
3080 RESTORE 3070
3090 FOR Z=1 TO 10
3100 READ Z1
3110 CALL HCHAR(22,Z+11,Z1)
3120 NEXT Z
3130 FOR Z=99 TO 101
3140 CALL HCHAR(11,Z-69,Z)
3150 NEXT Z
3160 DATA 106,100,101,102,98
,99,97,98,102,97,104
3170 RESTORE 3160
3180 FOR Z=1 TO 11
3190 READ Z1
3200 CALL HCHAR(17,Z+18,Z1)
3210 NEXT Z
3220 DATA 12,2,96,12,3,115,1
2,4,116,13,2,114,16,19,105,1
6,29,103,11,22,111
3230 DATA 11,23,112,23,22,11
1,23,23,112,24,23,113,18,23,
107,19,29,107
3240 DATA 24,24,128,24,25,12
9,24,26,130,24,27,131,10,24,
120,11,24,121,10,25,122,
11,25,123
3250 DATA 23,29,103,24,29,10
4
3260 RESTORE 3220
3270 FOR Z=1 TO 23
3280 READ A,B,C
3290 CALL HCHAR(A,B,C)
3300 NEXT Z
3310 FOR Z=9 TO 11
3320 CALL COLOR(Z,11,1)
3330 NEXT Z
3340 CALL COLOR(12,9,1)
```

```
3350 CALL COLOR(13,14,1)
3360 FOR Z=3 TO 8
3370 CALL COLOR(Z,1,16)
3380 NEXT Z
3390 RANDOMIZE
3400 H=INT((31-3+1)*RND)+3
3410 V=2
3420 A$="ENERGIE"
3430 GOSUB 5250
3440 GOSUB 5190
3450 CALL HCHAR(V-1,H-1,40)
3460 CALL HCHAR(V,H-1,41)
3470 CALL HCHAR(V-1,H,42)
3480 CALL HCHAR(V,H,43)
3490 CALL COLOR(2,6,1)
3500 CALL JOYST(1,X,Y)
3510 CALL KEY(1,K,S)
3520 IF K=18 THEN 5000
3530 IF X=4 THEN 3670
3540 IF X=-4 THEN 3800
3550 V=V+1
3560 GOSUB 5130
3570 CALL GCHAR(V,H-1,A)
3580 CALL GCHAR(V,H,B)
3590 IF (A>95)+(B>95)THEN 39
70
3600 CALL HCHAR(V-2,H-1,32,2
)
3610 CALL HCHAR(V-1,H-1,40)
3620 CALL HCHAR(V,H-1,41)
3630 CALL HCHAR(V-1,H,42)
3640 CALL HCHAR(V,H,43)
3650 GOSUB 5130
3660 GOTO 3500
3670 H=H+1
3680 IF H>31 THEN 3930
3690 V=V+1
3700 GOSUB 5130
3710 CALL GCHAR(V,H,A)
3720 CALL GCHAR(V,H-1,B)
3730 CALL GCHAR(V-1,H,C)
3740 IF (A>95)+(B>95)+(C>95)
THEN 3780
3750 CALL VCHAR(V-2,H-2,32,2
)
3760 CALL HCHAR(V-2,H-1,32)
3770 GOTO 3610
3780 H=H-1
3790 GOTO 3970
3800 H=H-1
3810 IF H<3 THEN 3950
3820 V=V+1
3830 GOSUB 5130
3840 CALL GCHAR(V,H-1,A)
3850 CALL GCHAR(V,H,B)
3860 CALL GCHAR(V-1,H-1,C)
3870 IF (A>95)+(B>95)+(C>95)
THEN 3910
3880 CALL VCHAR(V-2,H+1,32,2
)
3890 CALL HCHAR(V-2,H,32)
3900 GOTO 3610
3910 H=H+1
3920 GOTO 3970
3930 H=H-1
3940 GOTO 3550
3950 H=H+1
3960 GOTO 3550
3970 IF (A>127)+(B>127)THEN
4330
3980 V=V-1
3990 CALL HCHAR(V-1,H-1,135,
2)
4000 CALL HCHAR(V,H-1,135,2)
4010 FOR Z=0 TO 30
4020 CALL SOUND(-2000,-7,Z)
4030 NEXT Z
4040 CALL HCHAR(V,H-1,32,2)
4050 CALL HCHAR(V-1,H-1,32,2
)
4060 GOSUB 5130
4070 FOR Z1=1 TO 5
4080 FOR Z=800 TO 400 STEP -
25
4090 CALL SOUND(-2000,Z,8)
4100 NEXT Z
4110 FOR Z=1400 TO 1000 STEP
 -25
4120 CALL SOUND(-2000,Z,4)
4130 NEXT Z
4140 NEXT Z1
4150 CALL SOUND(-100,40000,3
0)
4160 FOR Z=1 TO 1000
4170 NEXT Z
4180 CALL CLEAR
4190 FOR Z=1 TO 16
4200 CALL COLOR(Z,16,1)
4210 NEXT Z
4220 PRINT "WOLLEN SIE ES NO
CHEINMAL"::"SPIELEN? DANN DR
UECKEN SIE"::"""BACK""":
::::
4230 CALL KEY(5,K,S)
4240 IF S=0 THEN 4230
4250 IF K=15 THEN 4270
4260 END
4270 FOR Z=1 TO 16
4280 CALL COLOR(Z,2,1)
4290 NEXT Z
4300 T=0
4310 EM=0
4320 GOTO 290
4330 GOSUB 5130
4340 CALL HCHAR(24,1,96,21)
4350 A$="GUT GELANDET"
4360 GOSUB 5250
4370 FOR Z1=1 TO 10
4380 FOR Z=800 TO 1200 STEP
50
4390 CALL SOUND(-2000,Z,6,Z+
25,10)
4400 NEXT Z
4410 NEXT Z1
4420 CALL SOUND(-200,40000,3
0)
4430 FOR Z=1 TO 1000
4440 NEXT Z
4450 IF T>13 THEN 5290
4460 CALL CLEAR
4470 FOR Z=1 TO 16
4480 CALL COLOR(Z,2,1)
4490 NEXT Z
4500 T=0
4510 GOTO 300
4520 H=H+1
4530 V=V-1
4540 CALL SOUND(-450,-5,16)
4550 IF (H<4)+(H>29)+(V<3)+(
V>22)THEN 4610
4560 CALL HCHAR(V+1,H-1,G)
4570 CALL GCHAR(V,H,G)
4580 GOSUB 2470
4590 IF G=96 THEN 4610
4600 CALL HCHAR(V,H,F)
4610 D=D+2
4620 IF D>=E THEN 2250
4630 GOTO 1260
4640 H=H-1
4650 V=V-1
4660 CALL SOUND(-450,-5,16)
4670 IF (H<4)+(H>29)+(V<3)+(
V>22)THEN 4730
4680 CALL HCHAR(V+1,H+1,G)
4690 CALL GCHAR(V,H,G)
4700 GOSUB 2470
4710 IF G=96 THEN 4730
4720 CALL HCHAR(V,H,F)
4730 D=D+2
4740 IF D>=E THEN 2250
4750 GOTO 1260
4760 H=H+1
4770 V=V+1
4780 CALL SOUND(-450,-5,16)
4790 IF (H<4)+(H>29)+(V<3)+(
V>22)THEN 4850
4800 CALL HCHAR(V-1,H-1,G)
4810 CALL GCHAR(V,H,G)
4820 GOSUB 2470
4830 IF G=96 THEN 4850
4840 CALL HCHAR(V,H,F)
4850 D=D+2
4860 IF D>=E THEN 2250
4870 GOTO 1260
4880 H=H-1
4890 V=V+1
4900 CALL SOUND(-450,-5,16)
4910 IF (H<4)+(H>29)+(V<3)+(
V>22)THEN 4970
4920 CALL HCHAR(V-1,H+1,G)
4930 CALL GCHAR(V,H,G)
4940 GOSUB 2470
4950 IF G=96 THEN 4970
4960 CALL HCHAR(V,H,F)
4970 D=D+2
4980 IF D>=E THEN 2250
4990 GOTO 1260
5000 IF V<3 THEN 3500
5010 IF EN=25 THEN 3550
5020 EN=EN-25
5030 GOSUB 5190
5040 GOSUB 5130
5050 CALL HCHAR(V-1,H-1,32,2
)
5060 CALL HCHAR(V,H-1,32,2)
5070 CALL SOUND(-600,110,25,
880,22,-6,20)
5080 V=V-1
5090 CALL GCHAR(V-1,H,XA)
5100 CALL GCHAR(V-1,H-1,XB)
5110 IF (XA>95)+(XB>95)THEN
5340
5120 GOTO 3610
5130 IF R=1 THEN 5160
5140 R=1
5150 GOTO 5170
5160 R=5
5170 CALL COLOR(2,R,1)
5180 RETURN
5190 CALL HCHAR(24,16,96,6)
5200 FOR Z=1 TO LEN(STR$(EN)
)
5210 CALL HCHAR(24,15+Z,ASC(
SEG$(STR$(EN),Z,1)))
5220 NEXT Z
5230 IF EN<=0 THEN 4070
5240 RETURN
5250 FOR Z=1 TO LEN(A$)
5260 CALL HCHAR(24,7+Z,ASC(S
EG$(A$,Z,1)))
5270 NEXT Z
5280 RETURN
5290 A$="GRATULIERE GEWONNEN
"
5300 GOSUB 5250
5310 FOR Z=1 TO 2000
5320 NEXT Z
5330 GOTO 4180
5340 V=V+1
5350 CALL HCHAR(V-1,H-1,135,
2)
5360 CALL HCHAR(V,H-1,135,2)
5370 GOTO 4010
```


2300 Kiel

MCC Laden
Micro Computer Christ
Rathausstraße 4, 2300 Kiel 1
Telefon (04 31) 9 63 76
APPLE ATARI BASIS Commodore DAI EACA OSBORNE SHARP TANDY

5000 Köln

BUCHHANDLUNG
GONSKI Fachbücher + Fachzeitschriften für Mikrocomputer
Gertrudenstraße 2–4 Ecke Neumarkt)
5000 Köln 1 Telefon (0221) 210528

3000 Hannover

Vertragshändler
IBM PERSONAL COMPUTER
EPSON
TA TRIUMPH-ADLER
HEWLETT PACKARD
Microcomputer Software · Drucker Literatur · EDV-Zubehör
trend
Der Computer-Laden
AM MARSTALL 20/22 · 3000 HANNOVER 1
☎ 0511/14629

6630 Saarlouis

6630 Saarlouis, Lothringerstraße 9
Minninger ELEKTRONIK FUNK

8500 Nürnberg

Micro-Computer, Peripherie und Software GmbH
MCPS
APPLE SHARP EPSON CENTRONICS FELTRON
IBS Interface SINCLAIR SOFTWAREERSTELLUNG
Gibitzenhofstr 69, 8500 Nurnberg 1, Tel. (09 11) 67 70 93

4000 Düsseldorf

IHR GROSSER PARTNER FÜR KLEINE COMPUTER
DATA BECKER
Merowingerstr 30 4000 Dusseldorf Tel 0211 310010

Charakter-String (CHR$) beim TI-99/4A

Vielen frischgebackenen Texas-Besitzern bereiten die verschiedenen String-Funktionen einige Schwierigkeiten und werden deshalb oft gar nicht genutzt. Das ist umso trauriger, als sich eine Menge damit machen läßt.

Mit der String-Funktion (CHR$) wird dasjenige Zeichen erzeugt, das dem angegebenen ASCII-Code entspricht. Sage ich zum Beispiel:

```
X = 65
PRINT CHR$(X)
```

so wird mit größter Wahrscheinlichkeit ein "A" ausgegeben, weil der (dezimale) ASCII-Code für "A" der Wert 65 ist.
Geben Sie nun mal in Ihren TI-99 das folgende ein:

```
10 X=8:Y=7:Z=15
20 PRINT CHR$(X+Y+Z)
```

Was erscheint auf dem Bildschirm?

Richtig, der Cursor, weil der dezimale ASCII-Code für dieses Sonderzeichen die 30 ist – und die Zahlen 8,7 und 15 ergeben zusammen eine 30.
Wir sehen also, daß in der Klammer nach dem CHR$ nicht unbedingt ein einzelner Wert, sondern ruhig auch eine Rechenanweisung stehen kann, die zunächst ausgeführt wird, bevor Ihr Ergebnis an den ASCII-Code "überwiesen" wird.
Nun ist es beileibe nicht so, daß stets "Zeichen" angezeigt werden, wenn man die CHR$-Funktion anwendet:
Geben Sie einmal dazu das folgende Miniprogramm ein, das Ihnen, wenn Sie alles richtig gemacht haben, die "Zeichen" des ASCII-Codes, sowie ihre dezimale Verschlüsselung angibt.

```
10 CALL KEY (O,T,ST)
20 IF ST=0 THEN 10
30 CODE=ASC(A$)
40 PRINT "ASCII-Zeichen", "DEZ.-CODE"
50 PRINT CHR$(T),T
60 GOTO 10
```

Wenn Sie diesen Sechszeiler gestartet haben, werden Sie feststellen, daß nicht jeder Dezimalcode ein Zeichen auf den Bildschirm bringt. So bleibt die Spalte "ASCII-Zeichen" beispielsweise frei, wenn unter "Dez.-Code" eine 32 angezeigt wird – denn die 32 steht für "SPACE", also eine Leerstelle.
Für unser kleines Programm von eben sind nicht alle ASCII-Codes von Interesse. Da die Werte 0 bis 29 Steuerzeichen für rechnerinterne Vorgänge sind, sind Sie für Bildschirmdarstellung unergiebig. Dem gegenüber liefern die Werte 30 bis 127 darstellbare "characters" und können mit dem Abfrageprogramm leicht erlernt werden.
Das soll aber nicht heißen, daß uns die Werte < 30 nichts angingen: auch unter ihnen gibt es Funktionen, die der Datenübertragung (beispielsweise zum Drucker) dienen und deshalb eine Beachtung wert sind.

engl. Abk.	deutsche Bedeutung	CHR$(...)	CONTROL
NUL	Stornierung	0	.
BEL	Signal	7	G
HT	Tabulator, horiz.	9	I
LF	Zeilenvorschub	10	J
VT	Tabulator, vert.	11	K
FF	Blattvorschub	12	L
CR	Wagenrücklauf	13	M
SO	Sperrschrift	14	N
SI	Engschrift	15	0
DC1	Peripherie x-on	17	Q
DC2	Peripherie-Adresse	18	R
DC3	Peripherie x-off	19	S
DC4	Peripherie-Adresse	20	T
CAN	Buffer löschen	24	X
ESC	Präfix-Zeichen	27	

Hier nun eine Zusammenstellung derjenigen Zeichen, die Sie vor allem als Besitzer eines Druckers benötigen werden. Wir erreichen sie entweder über unsere CHR$-Funktion oder aber mittels der "CONTROL"-Taste und eines speziellen Texascodes.
Die englischen Abkürzungen der linken Spalte stellen sogenannte "symbolic codes" dar, d.h. eine Direkteingabe ist bei ihnen nicht möglich: Tippen Sie zum Beispiel BEL ein, so wird kein Signalpiepser ertönen. Somit können wir die Abkürzungen als "mnemonics" (Gedächtniskürzel) auffassen.
Die Angabe "Präfix-Steuerzeichen" besagt, daß der ESC-Code von weiteren Steuerzeichen begleitet wird, diese also stets nur einleitet (Präfix= Vorsilbe).
Hier nun aber noch einige Zusatzinformationen zu einzelnen Steuerzeichen aus der oben stehenden Tabelle

BEL Bell:
Die meisten handelsüblichen Drucker besitzen als Signalgeber einen eingebauten Summer oder Beep-Generator. Wird vom TI nun ein BEL gesendet, so schaltet sich dadurch dieser Summer für etwa 3 Sekunden ein. Das ist praktisch, um einzelne Phasen des protokollierten Programmablaufes zu kennzeichnen, sei es, daß der Drucker stoppt, weil der Computer ein Input benötigt, sei es, weil das Papier alle ist.

LF Line Feed:
Wenn dieses Zeichen übertragen wird, so prüft der Drucker, ob sich ein Zeichen im Datenkanal befindet. Ist dies der Fall, so läßt er es passieren und führt danach LF aus (schiebt das Papier um eine Zeile weiter). Ist jedoch kein Zeichen mehr im Bus, so wird der Zeilenvorschub sogleich ausgeführt.

CR Carriage Return:
Was der Wagenrücklauf bedeutet, muß – so glaube ich – nicht besonders erläutert werden. Da die CR-Funktion drucktechnisch jedoch mehr tut, als den Druckkopf an den Zeilenanfang zurückzubringen, ist der Carriage Return für einige hübsche Spielereien brauchbar. CR bewirkt nämlich auch, daß alle im Buffer des Druckers befindlichen Zeichen ausgedruckt werden. Dadurch können wir ohne großen Aufwand Fettdruck erzeugen, wenn wir den CR-Befehl, sprich: **CHR$(13)**, geschickt in ein Programm einbauen.

```
10 OPEN#1:"RS232.BA=1200.DA=8"
20T$="FETTDRUCK"
30 PRINT#1:T$;CHR$(13);T$
```

In Zeile 10 wird die Ausgabe zum Drucker geregelt (Schnittstelle ist hierbei die RS-232 des TI-99. BA ist die Baudrate von 1200 und DA die Adresse).
In Zeile 20 wird ein Beispielstring angegeben, der in Zeile 30 ausgedruckt wird.
Der Witz besteht in dem CHR$(13)-Befehl, der zwischen zwei ";" gesetzt wurde und so die beiden T$-Anforderungen nahtlos miteinander verbindet.

FF Form Feed:
Stellt eine nützliche Variante des LF-Befehls dar. Der Unterschied besteht darin, daß nicht die nächste **Zeile**, sondern (bei Endlospapier mit Lochrand) das nächste **Blatt** angesteuert wird.

SO Shift Out:
Bringt gedehnte, also Sperrschrift, auf's Papier. Das klappt jedoch nur, wenn die jeweilige Druckzeile nicht mehr als 40 Zeichen umfaßt, da Leerstellen entstehen und auf die zulässige Gesamtzeilenlänge (im allgemeinen 80 Z.) angerechnet werden.

SI Shift In:
Ist das Gegenteil der oben genannten Funktion, nämlich ein engeres Schriftbild als normal. Die Buchstaben, Ziffern und übrigen Elemente des Texas-Zeichensatzes werden dicht gedrängt gedruckt, was sich zur optischen Gliederung von Texten sehr gut eignet.

Da "Character-String-Functions" sehr gut auch in Basic implementiert werden können, erweitern sie die Möglichkeiten des TI-Basic (bzw. Extended Basic) ebenso sinnvoll wie enorm. Gerade in Zusammenwirkung mit einem angeschlossenen Drucker sind dem leistungsstarken TI-99/4A somit neue Bereiche eröffnet.

LEASING-CONSULT

für den Sharp PC-1251
in Verbindung mit Drucker-Interface CE-125

Leasing wird in vielen Bereichen immer beliebter: Autos werden genauso geleast wie Farbfernsehgeräte, Stereo-Anlagen und Personalcomputer.

Noch vor wenigen Jahren hätte Leasen, also mieten, für äußerst unsolide gegolten: Heute arbeiten viele Großunternehmen ausschließlich auf dieser Basis.
Im allgemeinen unterscheidet man zwei Formen des Leasing: "financial" und "operational". Financial Leasing dient dem Erwerb einer relativ teuren Sache, ohne die gesamten Ersparnisse des Mieters auf einmal in Anspruch nehmen zu müssen. Dabei geht man von einem Restwert 0 aus – der Mieter wird nach der festgelegten Leasingzeit Eigentümer des betreffenden Gegenstandes, trägt aber auch während des Leasings alle Eventualkosten.
Operational Leasing hat von Anfang an gar nicht den Erwerb der gemieteten Sache durch den Mieter im Sinn, sondern nur die Nutzung durch ihn (klassisches Beispiel: die Wohnungsmiete). Selbstverständlich wird hier auch ein Restwert nach Ende der Mietzeit angenommen, der dem Verleiher wieder zur Verfügung steht.
Leasing bietet sich vor allem bei relativ kurzlebigen Artikeln an, um das Käuferrisiko so niedrig wie möglich zu halten: Beispielsweise werden heute Großrechenanlagen fast ausnahmslos geleast, damit die Firma, die den Großcomputer mietet, nicht nach kurzer Zeit dem technologischen Fortschritt hinterher hinkt.

Steht man vor der Wahl, ob man kaufen oder leasen soll, ist zu berücksichtigen, wie günstig ein Kauf zu realisieren wäre, wie preiswert z.B: ein Bankkredit ist.

Da bei derlei Überlegungen natürliche verschiedene Faktoren bedeutsam werden, ist ein kleines Programm wie das vorliegende äußerst nützlich: Nach Eingabe der wichtigen Daten (Anschaffungspreis, Zinssatz, Laufzeit, Steuersatz, Restwert usw.) trifft der PC-1251 die Entscheidung "Kaufen oder Leasen?" für Sie. Ist vielleicht schon etwas für den nächsten Autokauf?!

Variablenliste:
A: Abschreibebetrag pro Jahr
I: Zinssatz %
J: Laufzeit
L: Leasingbetrag pro Jahr
M: Mehrzweckvariable
P: Anschaffungspreis (Investition)
Q, Q$: Restwert nach J Jahren
S: Marginalsteuersatz %
T: Tilgung pro Jahr
X: Darlehen nach Zinsabzug
Y: Leasingbetrag nach Zinsabzug
Z: Mehrzweckvariable

Hinweis:
Das vorliegende Programm läßt sich durch Kürzen der auf Display ausgegebenen Texte leicht auf den PC-1245 umschreiben.

```
10:REM . LEASING-CONSULT
11:REM FUER DEN PC-1251
12:REM      (MIT CE-125)
13:REM
15:WAIT 0: PRINT "":
   CALL &1150
16:POKE 63488,0,38,73,7
   3,50,127,16,40,68
17:FOR Q=1 TO 30: NEXT
   Q
18:WAIT 0: PRINT ""
20:WAIT 150: PRINT "<DA
   TEN-EINGABE>     ---
   >"
21:INPUT "<INVESTITION>
   ? ";P
22:INPUT "<ZINSSATZ> (%
   ) ? ";I:I=I/100
23:INPUT "<LAUFZEIT> (J
   AHRE) ? ";J
24:INPUT "<RESTWERT> ?
   ";R
25:INPUT "<MARG.-STEUER
   SATZ> ? ";S:S=S/100
26:INPUT "<TILG./JAHR>
   ? ";T
27:INPUT "<LEAS./JAHR>
   ? ";L
28:X=0:Y=0
30:WAIT 100: PRINT "<AB
   SCHREIBUNGEN>    ---
   >"
31:FOR Z=1 TO J
32:PAUSE "IM ";Z;"JAHR"
33:INPUT A
34:M= ABS (P-(( INT (P*
   (1+I))*100+.5)/100))
35:M=(A+M)*S:M= INT (M*
   100+.5)/100
36:M=(T-M)/((1+I)^Z):A=
   INT (M*100+.5)/100
37:P=P-T+M
38:X=X+A:Y=(Y+ INT ((L-
   (L*S))/(1+I)^Z))
39:NEXT Z
40:X=X-( INT (R/(1+I)^J
   )*100+.5)/100
50:USING "##########.##"
60:LPRINT "<DARL.,ABGEZ
   INST>":  LPRINT X;"
   DM"
61:LPRINT "<LEAS.,ABGEZ
   INST> ": LPRINT Y;"
   DM"
62:IF Y<X THEN 80
70:GOSUB 96
71:GOSUB 94
72:GOSUB 97
73:GOSUB 95
74:GOSUB 98
75:LPRINT Y-X;"  DM"
76:GOTO 90
80:GOSUB 96
81:GOSUB 95
82:GOSUB 97
83:GOSUB 94
84:GOSUB 98
85:LPRINT X-Y;"  DM"
90:PRINT "<NEUER LAUF /
   ENDE>  N/E"
91:Q$= INKEY$ : IF Q$="
   " THEN 91
92:IF Q$="N" THEN 20
93:END
94:LPRINT "K A U F E N"
   : RETURN
95:LPRINT "L E A S E N"
   : RETURN
96:LPRINT "SOMIT IST":
   RETURN
97:LPRINT "GUENSTIGER A
   LS": RETURN
98:LPRINT "": LPRINT "G
   ESPART:": RETURN
```

Chicken Joke

für den TI-99/4A + Extended Basic

Versuchen Sie möglichst viele Eier aufzufangen, die Ihre Henne legt und fallen läßt!
Viele Raffinessen und Schwierigkeiten müssen bewältigt werden: das Spiel endet, wenn Sie 3 Eier nicht gefangen haben.
Mit steigender Rundenzahl (max. 4) ändert sich die Spielsituation, d.h. Sie bekommen es mit weiteren Hindernissen zu tun.
Von Runde zu Runde erhöht sich ebenfalls die Anzahl der zu fangenden Eier!
Also aufpassen! Ihre Henne wird immer fleißiger!
Für jedes gefangene Ei erhalten Sie 10 Punkte. Zerschellt ein Ei am Boden, so werden Ihnen Punkte abgezogen.
Gesteuert wird mit dem Joystick, Chicken-Joke kann einzeln oder zu zweit gespielt werden.
Es warten insgesamt 6 Levels auf Ihr Geschick.

Programmbeschreibung:
130-170: Variablen-Indizierung
180-340: Beginn der großen Spielschleife, Zeichen-Definitionen
350: Abfrage, ob Titelbild erscheinen soll
360-450: Bildaufbau - der jeweiligen Runde entsprechend
460-580: Kleine Spielschleife: Bewegung des Bauern mit verschiedenen Verzweigungen (Ei gefangen oder nicht gefangen, Mauer oder liegendes Ei berührt, oder dgl.). Diese Schleife wird RHY mal durchlaufen, dann wird ein neues Ei gelegt.
590-680: Huhn legt Ei mit zufälliger Fallrichtung
690-760: Spielerwechsel, wenn dieser notwendig ist
770-790: Einer der beiden Spieler scheidet aus
800-880: Punktezählung und Anzeige
890-980: Bauer stolpert über liegendes Ei
990-1070: "Game over" für einen der beiden Spieler
1080-1270: Titelbild
1280-1410: Endebild
1420-1640: Melodie

Variablenliste:
TI: Schaltung zum Titelbild
G: Soviele Eier müssen gefangen werden (Catch:...)
ROUND: Runde
SP: Spieler
BONUS: Gutschrift für gefangenes Ei
P(1), P(2): Punkte der beiden Spieler
OUT: Eier, die nicht gefangen wurden
IN: Eier, die gefangen wurden
VO: Geschwindigkeit des Bauern
RHY: Abstand, in welchem Eier gelegt werden
V: Spaltengeschwindigkeit der Eier, in Abhängigkeit vom Huhn
K: Spielschleifenzähler (kleine Spielschleife
M: Sprite bzw. Ei, das dem Boden am nächsten ist
Z: Geschwindigkeit des Huhns (zufällig)
U: Sprite bzw. Ei, das dem Huhn am nächsten ist
EOZ: Gibt die Anzahl der Spieler an
FLAG: Zeigt an, ob der Bauer nach links oder rechts fährt
I: Schleifenindex für diverse Schleifen
X, Y: Position des Joystick-Hebels
C, D: Position des Eies, das dem Boden am nächsten ist

```
100 !!!***CHICKEN-JOKE***!!!
110 ! COPYRIGHT BY E.HOEFLER
 AND D.BILLER (HB-VISION)
120 ! ADRESSE: E.HOEFLER, 80
56 NEUFAHRN, CARL-DIEM-STR.2
1, TEL.: 08165/4210
130 CALL CLEAR
140 CALL MAGNIFY(3)
150 RANDOMIZE
160 TI=0
170 VO,G=2 :: ROUND,SP=1 ::
BONUS=10 :: DIM P(2):: P(1),
P(2)=0
180 ! BEGINN DER SPIELSCHLEI
FE ************************
190 V,K,OUT,IN=0 :: U=5 :: M
=U
200 IF ROUND>20 AND((ROUND-1
)/4=INT((ROUND-1)/4))THEN VO
=VO-0.3
210 IF ROUND>8 AND ROUND<13
THEN RHY=11 ELSE RHY=13
220 IF ROUND>12 THEN RHY=11
230 CALL CHAR(96,"00181C26EC
1D0F0F0301000000000000000028
54BABB7DEFEFEFFE7C282828
6C")
240 CALL CHAR(100,"0000142A5
DDDBEF7F7F77F3E1414143600183
86437B8F8F0C080")
250 CALL CHAR(104,"020100000
0011148243D1F0100040E0F088C7
E0F9AD2FEDA041CFE1E3E1C0
8FC")
260 CALL CHAR(112,"10317EF05
94B7F5B20387F787C38103F40800
0000080881224BCF88000207
0F0")
270 CALL CHAR(136,"FFFF")
280 CALL CHAR(108,"000000000
1030707070F0F0F070300000000000
0000080C0C0E0E0E0C080")
290 CALL CHAR(127,"00FBFBFB0
07F7F7F")
300 CALL CHAR(116,"000000000
0000001835F3F074BE9FE6D00070
F0B183070190B3F79F99939F
B33")
310 CALL CHAR(120,"00E0F0D01
80C0E98D0FC9E9F999CDFCC00000
000000000080C1F2FCE0D2977
FB6")
320 CALL CHAR(128,"FFFFFFFFF
FFFFFFF")
330 CALL CHAR(130,"001800C30
0180000",131,"00C3000000C300
00")
340 CALL COLOR(13,16,02,14,9
,1,12,10,15)
350 IF TI=0 THEN GOSUB 1080
360 FOR I=2 TO 8 :: CALL COL
OR(I,16,2):: NEXT I
370 CALL HCHAR(3,1,136,32)
380 CALL HCHAR(24,1,127,32)
390 IF ROUND>12 THEN CALL VC
HAR(20,16,127,4)
400 IF (ROUND>4 AND ROUND<9)
OR ROUND>16 THEN CALL HCHAR(
10,1,128,192)
410 DISPLAY AT(4,1)SIZE(5):"
ROUND" :: DISPLAY AT(4,6)SIZ
E(3):ROUND
420 DISPLAY AT(6,1)SIZE(6):"
PLAYER" :: DISPLAY AT(6,7)SI
ZE(2):SP
430 DISPLAY AT(4,18)SIZE(7):
"POINTS:" :: DISPLAY AT(4,25
)SIZE(6):P(SP)
440 DISPLAY AT(6,18)SIZE(6):
"CATCH:" :: DISPLAY AT(6,24)
SIZE(2):G-IN
```

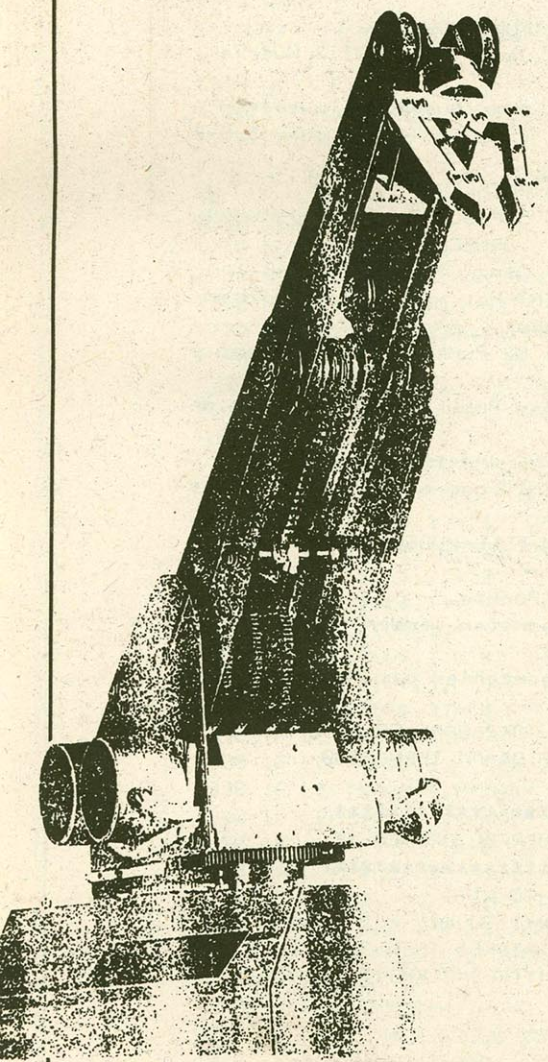
```
450 CALL SPRITE(#1,100,15,1,
100,#2,104,4,168,100)
460 ! BEWEGUNG BAUER *******
470 CALL JOYST(SP,X,Y)
480 IF X=4 AND Y=0 THEN CALL
 PATTERN(#2,112):: CALL MOTI
ON(#2,0,V0*X)
490 IF X=0 AND Y=0 THEN CALL
 MOTION(#2,0,0)
500 IF X=-4 AND Y=0 THEN CAL
L PATTERN(#2,104):: CALL MOT
ION(#2,0,V0*X)
510 K=K+1 :: IF K=RHY THEN G
OSUB 600
520 IF ROUND>8 AND ROUND<13
THEN CALL POSITION(#M,C,D)::
 IF C>165 THEN GOSUB 860
 ELSE 550
530 CALL POSITION(#M,C,D)
540 IF C>165 THEN CALL DELSP
RITE(#M):: CALL SOUND(100,-5
,0):: OUT=OUT+1 :: P(SP)
=P(SP)-(BONUS*G/2):: GOSUB 8
40 :: IF OUT=3 THEN 780
550 IF ROUND>8 AND ROUND<13
THEN CALL COINC(#2,#25,10,T)
:: IF T THEN 890
560 CALL COINC(#2,#M,8,T)::
IF T THEN GOSUB 810
570 IF ROUND>12 THEN CALL CO
INC(#2,168,117,10,T):: IF T
THEN CALL SOUND(270,-5,0
):: GOTO 780
580 GOTO 470
590 ! EI WIRD GELEGT *******
600 V=Z
610 CALL MOTION(#1,0,0):: CA
LL POSITION(#1,A,B)
620 Z=INT(RND*11)-5
630 U=U+1 :: M=U-2
640 IF Z>0 THEN CALL PATTERN
(#1,100)ELSE CALL PATTERN(#1
,96)
650 CALL MOTION(#1,0,Z)
660 CALL SPRITE(#U,108,16,A,
B,5,V)
670 K=0
680 RETURN
690 ! SPIELERWECHSEL *******
700 FOR I=1 TO 25 :: CALL MO
TION(#I,0,0):: NEXT I :: FOR
 I=1 TO 50 :: NEXT I
710 IF EOZ=1 THEN 740
720 IF SP=2 THEN SP=1 :: GOT
O 740
730 IF SP=1 THEN SP=2 :: CAL
L CLEAR :: CALL DELSPRITE(AL
L):: GOTO 190
740 G=G+1 :: CALL CLEAR :: C
ALL DELSPRITE(ALL)
750 ROUND=ROUND+1 :: IF (ROU
ND-1)/4=INT((ROUND-1)/4)THEN
 G=2
760 GOTO 190
770 ! EIN SPIELER SCHEIDET A
US ************************
780 IF SP=1 THEN SP=2 ELSE S
P=1
790 IF EOZ=1 THEN 1280 ELSE
EOZ=1 :: GOSUB 990 :: CALL C
LEAR :: CALL DELSPRITE(A
LL):: GOTO 190
800 ! PUNKTEZAEHLUNG *******
810 CALL SOUND(250,700,0)::
CALL DELSPRITE(#M,#25):: IN=
IN+1 :: P(SP)=P(SP)+BONU
S
820 IF IN=G THEN 700
830 DISPLAY AT(6,18)SIZE(6):
"CATCH:" :: DISPLAY AT(6,24)
SIZE(2):G-IN
840 DISPLAY AT(4,18)SIZE(7):
"POINTS:" :: DISPLAY AT(4,25
)SIZE(6):P(SP)
850 RETURN
860 CALL DELSPRITE(#M):: CAL
L SPRITE(#25,108,16,175,D)
870 OUT=OUT+1 :: P(SP)=P(SP)
-(BONUS*G/2):: GOSUB 840 ::
IF OUT=3 THEN 780
880 RETURN
890 !BAUER GESTOLPERT ******
900 IF X=-4 THEN FLAG=-1 ELS
E IF X=4 THEN FLAG=1
910 CALL SOUND(500,200,0)
920 CALL MOTION(#2,0,0)
930 IF FLAG=-1 THEN CALL PAT
TERN(#2,116)ELSE CALL PATTER
N(#2,120)
940 CALL DELSPRITE(#25)
950 FOR I=1 TO 26 :: CALL MO
TION(#I,0,0):: NEXT I
960 CALL POSITION(#2,L,M)
970 FOR I=1 TO 50 :: CALL HC
HAR(INT(L/8),INT(M/8)+2,130,
1):: CALL HCHAR(INT(L/8)
,INT(M/8)+2,131,1):: NEXT I
980 GOTO 780
990 ! GAME OVER ************
1000 CALL CLEAR :: CALL DELS
PRITE(ALL)
1010 CALL SCREEN(5)
1020 DISPLAY AT(8,10):"GAME
OVER"
1030 DISPLAY AT(11,10):"PLAY
ER" :: IF SP=1 THEN DISPLAY
AT(11,17):"2" ELSE DISPL
AY AT(11,17):"1"
1040 FOR I=1 TO 900 :: NEXT
I
1050 IF SP=1 THEN ROUND=ROUN
D+1 :: G=G+1 :: IF (ROUND-1)
/4=INT((ROUND-1)/4)THEN
G=2
1060 CALL SCREEN(2)
1070 RETURN
1080 ! TITELBILD ************
1090 CALL SCREEN(7)
1100 FOR I=1 TO 8 :: CALL CO
LOR(I,2,16):: NEXT I
1110 DISPLAY AT(6,6)SIZE(19)
:"HB-VISION PRESENTS:"
1120 GOSUB 1430
1130 CALL SPRITE(#1,112,13,9
5,50,0,9)
1140 CALL SOUND(4250,-4,0)
1150 CALL POSITION(#1,A,B)
1160 IF B=200 THEN CALL MOTI
ON(#1,0,0):: CALL PATTERN(#1
,104):: GOTO 1180
1170 GOTO 1150
1180 DISPLAY AT(13,8)SIZE(12
):"CHICKEN-JOKE" :: DISPLAY
AT(14,8)SIZE(12):"_______
_______"
1190 DISPLAY AT(20,18)SIZE(4
):"1984"
1200 CALL SPRITE(#2,100,5,80
,90)
1210 FOR I=1 TO 1000 :: NEXT
 I
1220 CALL CLEAR :: CALL DELS
PRITE(ALL)
1230 DISPLAY AT(10,4):"ONE O
R TWO PLAYER ?"
1240 ACCEPT AT(10,24)BEEP VA
LIDATE("12"):EOZ :: IF EOZ>2
 THEN 1240
1250 CALL CLEAR :: CALL COLO
R(1,2,1):: FOR I=2 TO 8 :: C
ALL COLOR(I,16,2):: NEXT
 I
1260 CALL SCREEN(2):: TI=1
1270 RETURN
1280 ! ENDE-BILD ************
1290 CALL CLEAR :: CALL DELS
PRITE(ALL)
1300 FOR I=2 TO 8 :: CALL CO
LOR(I,2,1):: NEXT I
1310 CALL SCREEN(11)
1320 DISPLAY AT(5,3)SIZE(8):
"PLAYER 1" :: DISPLAY AT(5,2
0)SIZE(8):"PLAYER 2"
1330 DISPLAY AT(10,12)SIZE(6
):"POINTS"
1340 DISPLAY AT(10,3)SIZE(6)
:P(1):: DISPLAY AT(10,20)SIZ
E(6):P(2)
1350 IF P(1)>P(2)THEN CALL S
PRITE(#1,112,7,100,45,#2,116
,7,100,180)ELSE CALL SPR
ITE(#3,120,7,100,45,#4,104,7
,100,180)
1360 DISPLAY AT(20,3)SIZE(15
):"NEW GAME? (Y/N)"
1370 GOSUB 1430
1380 CALL KEY(0,K,S):: IF S=
0 THEN 1380
1390 IF K=89 OR K=121 THEN C
ALL CLEAR :: CALL DELSPRITE(
ALL):: GOSUB 1230 :: CAL
L CLEAR :: CALL SCREEN(2)::
GOTO 170
1400 IF K=78 OR K=110 THEN E
ND
1410 GOTO 1380
1420 ! TITELMELODIE ********
1430 GOSUB 1620
1440 FOR I=1 TO 50 :: NEXT I
1450 GOSUB 1630
1460 FOR I=1 TO 100 :: NEXT
I
1470 GOSUB 1620
1480 FOR I=1 TO 50 :: NEXT I
1490 CALL SOUND(180,300,0)::
 CALL SOUND(190,385,0):: CAL
L SOUND(190,390,0):: CAL
L SOUND(240,340,0)
1500 FOR I=1 TO 100 :: NEXT
I
1510 GOSUB 1620
1520 FOR I=1 TO 50 :: NEXT I
1530 GOSUB 1630
1540 FOR I=1 TO 50 :: NEXT I
1550 GOSUB 1640
1560 GOSUB 1640
1570 CALL SOUND(140,350,0)::
 CALL SOUND(200,330,0):: CAL
L SOUND(260,270,0)
1580 FOR I=1 TO 40 :: NEXT I
1590 CALL SOUND(180,270,0)::
 CALL SOUND(100,40000,30)::
CALL SOUND(180,270,0)::
FOR I=1 TO 40 :: NEXT I
1600 CALL SOUND(100,270,0)::
 CALL SOUND(100,270,0):: FOR
 I=1 TO 30 :: NEXT I ::
CALL SOUND(230,270,0)
1610 RETURN
1620 CALL SOUND(120,290,0)::
 CALL SOUND(120,265,0):: CAL
L SOUND(120,240,0):: CAL
L SOUND(120,215,0):: RETURN
1630 CALL SOUND(180,300,0)::
 CALL SOUND(300,385,0):: CAL
L SOUND(140,390,0):: CAL
L SOUND(200,340,0):: CALL SO
UND(200,270,0):: RETURN
1640 CALL SOUND(140,350,0)::
 CALL SOUND(200,330,0):: CAL
L SOUND(200,270,0):: RET
URN
```

★★★★★

Aus der beiliegenden Fotokopie ersehen Sie einen Mini-Industrieroboter.
Da Sie bestimmt laufend Informationen über neue Peripheriegeräte erhalten, sollten Sie mir sicher Auskunft geben können, wo ich mir Unterlagen über solche Roboter beschaffen könnte.

S. Bianchetti, Nidau (CH)

Redaktion:
Da die Redaktion das vorliegende Modell nicht exakt identifizieren kann, gibt sie die Frage an unsere Leser weiter. Wir merken nur folgendes an: Der Roboter-Markt expandiert stetig. Für fast alle Rechner werden über kurz oder lang teil nützliche, teils Spielzeug-ähnliche Geräte zur Verfügung stehen. Wir sind sicher, daß Ihr Fachhändler schon bald einiges vorzeigen kann.



Zum besseren Verständnis der recht komplizierten Graphikprogrammierung des C-64 kaufte ich mir das DATA-BECKER-Graphikbuch. Leider wird in diesem Buch nur theoretisch eine Zeichenssatzveränderung in Multicolour beschrieben, aber kein praktisches Beispiel aufgeführt. Darum habe ich folgende Bitte:
Könnten Sie ein kleines Beispiel für die Zeichensatzveränderung (Multicolour) für den C-64 abdrucken.

T. Hackenberg, Lahr

Redaktion:
Viele solcher Zuschriften erreichten uns in den letzten Wochen. Ein Programm, in dem Zeichensatzveränderung mit Multicolour vorkommt, haben wir bereits in der CPU 5/84, Seite 20 abgedruckt.
Es handelt sich hierbei um das Programm Castle, das angesprochene Problem wird dort eindrucksvoll demonstriert.

Ich besitze einen TI-99/4A und stehe vor einem Rätsel.
Wie kann ich mit meinem TI in Extended Basic einen Kreis ziehen?
Ich würde mich freuen, wenn Sie mir helfen könnten.

D. Guter

Redaktion:
Setzen Sie das folgende kleine Listing in das Programm "Punkte plotten" aus CPU 11/84 ein und der TI-99 wird Ihnen alsbald einen schönen Kreis zeichnen.

```
1000: Platz für eigene
1010: Anwendung
1020:
1025: CALL CLEAR :: FOR I=34
TO 143 :: CALL CHAR (I," ") ::
NEXT I
1030: FOR I=1 TO 360
1040: X=INT (SIN(I)*50+100
1050: Y=INT (COS(I)*50+100
1060: CALL PLOTTEN (X.Y)
1070: NEXT I
1080:
1090:
```

Zu Ihrem Brief "Kopierer contra Paragraphen" in CPU 8/84 möchte ich anfragen, ob Ihre Redaktion und die Ihrer Konkurrenz-Zeitschriften nicht angesichts der weitgehend jugendlichen Leserschar – nicht von ungefähr reden viele Computer-Zeitschriften ihre Leser mit "Du" an – eine moralische Verantwortung verspüren und vor der Veröffentlichung von Gelegenheitsanzeigen, in denen entsprechende Software ohne Copyright angeboten wird, die Inserenten (und Kunden!) auf eventuelle juristische Konsequenzen hinweisen.

H. Hermann, Nürnberg

Redaktion:
Ihre Frage nach moralischer Verantwortung ist eigentlich angesichts Ihres Verweises auf unseren Artikel "Kopierer Contra § §" gegenstandslos. Besäßen wir für die von Ihnen angeschnittene Problematik keine "Antenne", so hätten wir sicher nicht so ausführlich dazu Stellung genommen. Lesen Sie doch bitte auch unsere Ausführungen im September-Heft (CPU 9/84), die wir unter der Überschrift "Die Geschichte vom Ausweg" veröffentlichten.
Dennoch ist Ihr Vorschlag, einen entsprechenden Hinweis auf jeder Seite "Kleinanzeigen" zu bringen, durchaus bemerkenswert. Man muß allerdings berücksichtigen, daß derartige pauschalierte Warnungen mit der Zeit ohnehin überlesen werden. Denken Sie nur an die Nikotin-Hinweise des Gesundheitsministeriums auf jeder Zigarettenschachtel, die den Absatz nicht haben beeinflußen können.

Seit Juni kaufe ich regelmäßig CPU und Homecomputer. In CPU 9 haben Sie ein verlockendes Angebot. Ich frage Sie nun:
1) Kann man auch CPU und Homecomputer Abonnement fürs Ausland abschließen?
2) Bekomme ich auch die Spielkassette geschickt?

R. Gasser, Hundling (F)

Redaktion:
Selbstverständlich gelten die Abonnement-Angebote auch für unsere Leser im Ausland – mit allen Vorteilen sowie der von Ihnen angesprochenen Kassetten-Prämie. Abonnement-Preise finden Sie in unserem Impressum.

Olympics

Ein Spiel für den TI 99/4A mit Extended Basic

Dieses Spiel ist für den **TI 99/4A** mit **Extended Basic** geschrieben. Gespielt wird mit der Tastatur.
Nach dem Eintippen oder Laden des Programms wird es mit „RUN" gestartet. Einige Sekunden später erscheint das Titelbild, und die Anzahl der Spieler muß eingegeben werden. Bis zu drei Spieler können teilnehmen. Danach sind die Spielernamen einzugeben. Hierbei ist zu beachten, daß die Namen der Spieler genau drei Buchstaben lang sind, da sonst das Programm an diesem Punkt nicht fortgesetzt wird.
Bevor der Programmverlauf weiter beschrieben wird, noch einige Erläuterungen. Bei allen **vier Disziplinen** ist es erforderlich, einmal zu laufen. Sie beschleunigen Ihre Figur, indem Sie abwechselnd zwei beliebige Tasten (außer der Space-Taste) im richtigen Takt niederdrücken. Sie werden den richtigen Rhythmus schnell herausbekommen, wenn Sie die Geschwindigkeitsanzeige links unten beobachten.
Bei jeder Disziplin ist eine große Anzeigetafel eingeblendet. Auf ihr befindet sich links oben der Punktestand nach der letzten Disziplin, rechts oben der Rekord. Links sind die erreichten Zeiten bei den Laufwettbewerben bzw. der beste Versuch beim Speerwurf oder Weitsprung. Rechts ist der aktuelle Spieler ausgedruckt. Hinter „QUALIFY" steht die Zeit bzw. Weite, die Sie unter- bzw. überbieten müssen, um weiterzukommen.

1. Disziplin: 100-m-Sprint

Ein Ton ist das Signal für den jeweiligen Spieler, sich für den Lauf bereitzuhalten. Nach dem Startschuß müssen Sie loslaufen, d. h., so lange zu „huddeln", bis Sie das Zielband durchlaufen haben. Sollten Sie schon vor dem Startschuß „huddeln", gibt es einen Fehlstart. Beim dritten Fehlstart scheiden Sie aus. Haben alle Teilnehmer den 100-m-Lauf absolviert, so kommen Sie zur **2. Disziplin, dem Weitsprung.** Hier müssen Sie nach Erklingen des Aufforderungstones los-„huddeln". Nach einer Anlaufzeit sehen Sie die Sprunggrube auftauchen. Sind Sie nahe genug am Sprungbalken, so springen Sie, durch Drücken der Space-Taste, ab. Je länger Sie sie gedrückt halten, desto größer der Absprungwinkel.
Für die **dritte Disziplin,** den **110-m-Hürdenlauf,** gilt im wesentlichen das gleiche wie für den 100-m-Lauf. Allerdings bleibt hier Ihre Figur, relativ gesehen, stehen und die Hürde bewegt sich auf Sie zu. Über die Hürde springen Sie, indem Sie die Space-Taste solange gedrückt halten, bis Sie über die Hürde hinweg sind.

Nun sind Sie bei der **vierten Disziplin,** dem **Speerwerfen,** angelangt. „Huddeln" Sie nach der akustischen Aufforderung bis kurz vor das Rasenfeld. Der Speer wird mit der Space-Taste abgeworfen. Je länger sie gedrückt wird, desto größer der Abwurfwinkel.

Sollten Sie bis jetzt noch nicht ausgeschieden sein, so beginnt das Spiel wieder beim 100-m-Lauf. Allerdings mit einer verkleinerten Qualifikationszeit. Insgesamt werden die Qualifikationsbedingungen fünfmal verschärft, dann bleiben sie konstant.

Sind alle Teilnehmer ausgeschieden, werden die besten acht Ergebnisse aufgelistet. Die Liste bleibt so lange bestehen, bis irgendeine Taste gedrückt wird. Danach erscheint wieder das Titelbild.

```
1 ! (C) 1984 BY MICHAEL HENN/6951 BILLIGHEIM
10 OPTION BASE 1 :: ON WARNING NEXT :: CALL CLEAR :: CALL MAGNIFY(3):: CALL SCRE
EN(2)
20 DIM SL$(3),WT(4),WT$(4),P(3),FO(3),ER(3,4),SP(7),P$(8),PL(8)
21 DATA 9.59,8.17,12.5,88.01
22 R1=1 :: FOR I=1 TO 4 :: READ WT(I):: NEXT I
30 IMAGE "##.##>"
31 IMAGE "##.##="
32 IMAGE "####<=/>        ##.##>"
40 FOR I=0 TO 8 :: CALL COLOR(I,16,1):: NEXT I :: DISPLAY AT(3,6):"O L Y M P I C
 S" :: DISPLAY AT(17,3):"(C) 1984 BY MICHAEL HENN" :: DISPLAY AT(10,2):"HOW MANY
 PLAYERS? (1-3)"
50 ACCEPT AT(10,26)SIZE(1)VALIDATE("123"):AN :: DISPLAY AT(5,6)ERASE ALL:"ENTER
YOUR NAMES!" :: MAN=AN
60 FOR J=1 TO AN :: I=J*2+8
70 ACCEPT AT(I,7)VALIDATE(UALPHA,NUMERIC):SL$((I-8)/2):: IF LEN(SL$((I-8)/2))<>3
 THEN CALL HCHAR(I,1,32,160):: GOTO 70
80 P(J),ER(J,4)=0 :: NEXT J
100 R=1 :: GOSUB 10000 :: FOR R3=1 TO AN :: V=10 :: W,T=0 :: W1=90 :: DISPLAY AT
(5,21)SIZE(6):USING 30:WT(1)
115 IF ER(R3,4)=-1 THEN 240
120 DISPLAY AT(10+R3,3)SIZE(3):SL$(R3):: FOR I=1 TO FO(R3):: CALL SPRITE(#32-R3*
3-I,92,16,73+R3*8,57+I*8):: NEXT I :: DISPLAY AT(11,19)SIZE(3):SL$(R3)
130 CALL SPRITE(#1,100,2,157,20,#2,36,1,161,175,#3,120,8,157,35,#4,132,16,161,51
):: DISPLAY AT(9,20)SIZE(6):USING 30:Q
```

```
140 CALL SOUND(500,440,5,-1,5):: FOR I=1 TO 750 :: NEXT I :: CALL SPRITE(#1,96,2
,157,35)
150 FOR I=1 TO 100 :: CALL KEY(0,K,S):: IF S THEN 250
160 NEXT I :: CALL SOUND(10,-7,0)
170 CALL KEY(0,K,S):: IF S THEN 180 ELSE T=T+.3 :: DISPLAY AT(24,18):USING 30:T*
.1 :: GOTO 170
180 CALL DELSPRITE(#3,#4)
190 CALL PATTERN(#1,100):: T=T+1.23 :: CALL KEY(0,K,S):: IF S=1 THEN V=40-(40-V)
*A ELSE V=40-(40-V)*B :: IF V<3 THEN V=3
200 CALL PATTERN(#1,104):: CALL MOTION(#1,0,V-2.2):: W=W+V*.04 :: DISPLAY AT(24,
3):USING "#####<=/>        ##.##>":V*37.5,T*.1 :: IF W<W1 THEN 190
210 IF W1=100 THEN 220 ELSE W1=100 :: CALL COLOR(#2,16):: GOTO 190
220 CALL MOTION(#1,0,0):: CALL PATTERN(#2,40)
230 CALL POSITION(#1,Y,X):: ER(R3,1)=INT(100*(T/10)-X+230)/100 :: DISPLAY AT(24,
18):USING 30:ER(R3,1):: DISPLAY AT(10+R3,11)SIZE(6):USING 30:ER(R3,1)
235 P(R3)=P(R3)+(20-ER(R3,1)+FO(R3))*105 :: IF P(R3)<0 THEN P(R3)=0
236 IF ER(R3,1)>0 THEN ER(R3,4)=-1 :: MAN=MAN-1
237 CALL WR(R,SL$(R3),WT(),WT$(),ER(R3,1)):: CALL TI
240 NEXT R3 :: IF MAN THEN 500 ELSE 3000
250 FO(R3)=FO(R3)-1 :: CALL COLOR(#31-R3*3-FO(R3),10):: DISPLAY AT(13,19)SIZE(4)
:"FOUL" :: CALL SOUND(400,-3,3):: FOR I=1 TO 300 :: NEXT I
260 DISPLAY AT(13,19)SIZE(4):"    " :: IF FO(R3)THEN IF R=1 THEN 130 ELSE 1020
270 ER(R3,4)=-1 :: MAN=MAN-1 :: GOTO 240
500 R=2 :: GOSUB 10000 :: CALL Z(AN,ER(,),SL$())
510 FOR R2=1 TO 3 :: FOR R3=1 TO AN :: IF ER(R3,4)THEN 800
520 W1=37 :: W,MA=0 :: V=10 :: CALL Z1(Q,WT(2),SL$(R3),R2,R3,MA,ER(,))
560 CALL SPRITE(#1,100,2,157,35,#4,132,1,161,143):: CALL SOUND(500,440,4,-1,7)::
 FOR I=1 TO 500 :: CALL KEY(0,K,S):: IF S THEN 580
570 NEXT I :: GOTO 900
580 CALL PATTERN(#1,100):: CALL KEY(0,K,S):: IF K=32 THEN 650 ELSE IF S=1 THEN V
=40-(40-V)*A ELSE V=40-(40-V)*B :: IF V<3 THEN V=3
590 CALL PATTERN(#1,104):: CALL MOTION(#1,0,V):: W=W+V*.04 :: DISPLAY AT(24,3)SI
ZE(8):USING "#####<=/>":V*37.5 :: IF W<W1 THEN 580
600 IF W1=37 THEN DISPLAY AT(21,17):A$ :: DISPLAY AT(22,17):B$ :: DISPLAY AT(23,
17):C$ :: CALL HCHAR(22,31,119,2):: W1=46 :: CALL COLOR(#4,16):: GOTO 580
610 CALL MOTION(#1,0,0):: GOTO 900
650 CALL MOTION(#1,0,0):: FOR PH=20 TO 80 :: CALL KEY(0,K,S):: IF K<>32 THEN 670
660 NEXT PH
670 DISPLAY AT(24,19)SIZE(3):USING "##;":PH :: CALL PATTERN(#1,108):: CALL POSIT
ION(#1,Y,X):: VY=SIN(PH/180*PI)*V :: VX=SQR(V*V-VY*VY)
680 CALL PATTERN(#1,112):: CALL SPRITE(#3,140,2,159,X):: FOR I=-VY/2 TO VY/3 ::
CALL MOTION(#1,I,VX):: NEXT I :: CALL MOTION(#1,0,0)
690 CALL POSITION(#1,Y1,X1):: CALL LOCATE(#1,161,X1):: W=VX*VY/85 :: FOR I=1 TO
90 :: NEXT I :: CALL SPRITE(#2,140,2,159,X1)
700 CALL SPRITE(#1,100,2,159,X1+8):: IF X>139 OR W1=37 THEN 900
710 ER(R3,R2)=INT((W-(139-X)/8)*100+.5)/100
750 CALL Z2(MA,ER(R3,R2),R2,R3):: CALL WR(R,SL$(R3),WT(),WT$(),ER(R3,R2))
770 IF R2=3 THEN P(R3)=P(R3)+MA*140 :: IF MA<0 THEN ER(R3,4)=-1 :: MAN=MAN-1
780 CALL TI :: CALL DELSPRITE(#1,#2,#3,#4):: CALL HCHAR(21,1,119,96):: CALL HCHA
R(22,1,128,32)
800 NEXT R3 :: NEXT R2 :: IF MAN THEN 1000 ELSE 3000
900 ER(R3,R2)=-1 :: DISPLAY AT(11+R2,21)SIZE(5):" FOUL" :: IF MA=0 THEN DISPLAY
AT(10+R3,11)SIZE(6):" FOUL"
910 IF R=2 THEN 770 ELSE 1770
1000 R=3 :: GOSUB 10000 :: FOR R3=1 TO AN :: V=10 :: W,T=0 :: W1=105 :: DISPLAY
AT(5,21)SIZE(6):USING 30:WT(3)
1005 IF ER(R3,4)=-1 THEN 1400
1010 DISPLAY AT(10+R3,3)SIZE(3):SL$(R3):: FOR I=1 TO FO(R3):: CALL SPRITE(#32-R3
*3-I,92,16,73+R3*8,57+I*8):: NEXT I :: DISPLAY AT(11,19)SIZE(3):SL$(R3)
1020 CALL SPRITE(#1,100,2,157,40,#3,120,8,157,55,#4,132,16,161,71,#2,108,1,161,2
56):: DISPLAY AT(9,20)SIZE(6):USING 30:Q
1030 CALL SOUND(500,440,4,-1,5):: FOR I=1 TO 750 :: NEXT I :: CALL SPRITE(#1,96,
2,157,55)
1040 FOR I=1 TO 100 :: CALL KEY(0,K,S):: IF S THEN 250
1050 NEXT I :: CALL SOUND(10,-7,0)
1060 CALL KEY(0,K,S):: IF S THEN 1070 ELSE T=T+.3 :: DISPLAY AT(24,18):USING 30:
T*.1 :: GOTO 1060
```

```
1070 CALL DELSPRITE(#3,#4):: CALL COLOR(#2,15)
1080 CALL PATTERN(#1,100):: CALL COINC(#1,#2,8,Y):: T=T+1 :: CALL KEY(0,K,S):: I
F K=32 THEN 1200 ELSE IF S=1 THEN V=40-(40-V)*A ELSE V=40-(40-V)*B :: IF V<3 THE
N V=3
1090 CALL PATTERN(#1,104):: CALL MOTION(#2,0,-V/1.5):: W=W+V*.04 :: CALL COINC(#
1,#2,8,X):: IF X=-1 OR Y=-1 THEN 1250 :: DISPLAY AT(24,3):USING 32:V*37.5,T*.1 :
: IF W<W1 THEN 1080
1100 CALL PATTERN(#2,36)
1105 IF W>130 THEN 1130
1110 CALL PATTERN(#1,100):: CALL KEY(0,K,S):: CALL COINC(#1,#2,8,Y):: IF S=1 THE
N V=40-(40-V)*A ELSE V=40-(40-V)*B :: IF V<3 THEN V=3
1120 CALL PATTERN(#1,104):: T=T+.9 :: DISPLAY AT(24,3):USING 32:V*37.5,T*.1 :: C
ALL COINC(#1,#2,8,X):: CALL MOTION(#2,0,-V/1.5):: W=W+V*.04 :: IF X=0 AND Y=0 TH
EN 1105
1130 CALL MOTION(#2,0,0):: CALL PATTERN(#2,40):: CALL DISTANCE(#1,#2,X):: X=SQR(
X):: T=INT(T*10-X+6)/100 :: ER(R3,1)=T :: GOTO 1300
1200 IF Y THEN 1250 ELSE CALL MOTION(#1,-50,0):: CALL PATTERN(#1,140):: CALL MOT
ION(#1,0,0)
1205 T=T+.3 :: CALL KEY(0,K,S):: IF K=32 THEN 1205 ELSE CALL LOCATE(#1,157,55)::
 GOTO 1090
1250 CALL PATTERN(#2,112):: CALL SOUND(-100,-7,0):: V=V*.8 :: FOR I=1 TO 25 :: N
EXT I :: T=T+4 :: CALL PATTERN(#2,108):: GOTO 1080
1300 DISPLAY AT(24,18)SIZE(6):USING 30:T :: DISPLAY AT(10+R3,11)SIZE(6):USING 30
:T :: W=(22-T+FO(R3))*105 :: IF W<0 THEN W=0
1310 P(R3)=P(R3)+W :: CALL WR(R,SL$(R3),WT(),WT$(),T):: CALL TI :: IF ER(R3,1)>0
 THEN ER(R3,4)=-1 :: MAN=MAN-1
1400 NEXT R3 :: IF MAN THEN 1500 ELSE 3000
1500 R=4 :: GOSUB 10000 :: CALL COLOR(10,16,4,11,16,4):: CALL Z(AN,ER(,),SL$())
1510 FOR R2=1 TO 3 :: FOR R3=1 TO AN :: IF ER(R3,4)THEN 1900
1520 W1=45 :: W,MA,MA1=0 :: V=10 :: CALL Z1(Q,WT(4),SL$(R3),R2,R3,MA,ER(,))
1530 CALL SPRITE(#1,96,12,157,29,#2,100,2,157,35):: CALL SOUND(500,440,4,-1,7)::
 FOR I=1 TO 500 :: CALL KEY(0,K,S):: IF S THEN 1550
1540 NEXT I :: GOTO 900
1550 CALL PATTERN(#2,100):: CALL KEY(0,K,S):: IF K=32 THEN 1650 ELSE IF S=1 THEN
 V=40-(40-V)*A ELSE V=40-(40-V)*B :: IF V<3 THEN V=3
1590 CALL PATTERN(#2,104):: CALL MOTION(#2,0,V-2,#1,0,V-2):: W=W+V*.04 :: DISPLA
Y AT(24,3)SIZE(8):USING "####<=/>":V*37.5 :: IF W<W1 THEN 1550
1600 IF W1=45 THEN DISPLAY AT(22,1):SEG$(A$,1,28):: CALL HCHAR(22,31,118,2):: W1
=57 :: GOTO 1550
1610 CALL MOTION(#2,0,0,#1,0,0):: GOTO 900
1650 CALL MOTION(#2,0,0,#1,0,0):: CALL POSITION(#2,Y,X):: IF X>136 OR W1=45 THEN
 900
1655 FOR PH=20 TO 80 :: CALL KEY(0,K,S):: IF K<>32 THEN 1670
1660 NEXT PH
1670 DISPLAY AT(24,19)SIZE(3):USING "##;":PH :: VY=SIN(PH/180*PI)*V :: VX=SQR(V*
V-VY*VY)
1680 CALL HCHAR(22,1,118,2):: IF PH>50 THEN CALL PATTERN(#1,36)ELSE IF PH>40 THE
N CALL PATTERN(#1,40)
1690 CALL DELSPRITE(#2):: Y=1 :: FOR I=-VY/3 TO VY/3 :: CALL MOTION(#1,I,0):: FO
R J=1 TO 5 :: Y=Y+VX/26.5 :: DISPLAY AT(22,1):SEG$(A$,INT(Y),28):: NEXT J
1700 J=I/V*3 :: IF J<-.8 THEN C=1 ELSE IF J<-.63 THEN C=2 ELSE IF J<-.3 THEN C=3
 ELSE IF J<.3 THEN C=4 ELSE IF J<.63 THEN C=5 ELSE IF J<.8 THEN C=6 ELSE C=7
1710 CALL PATTERN(#1,SP(C)):: NEXT I :: CALL POSITION(#1,I,J):: CALL MOTION(#1,0
,0):: CALL LOCATE(#1,159,J):: ER(R3,R2)=INT((.9*Y-(136-X)/8)*100)/100
1750 CALL Z2(MA,ER(R3,R2),R2,R3):: CALL WR(R,SL$(R3),WT(),WT$(),ER(R3,R2))
1770 IF R2=3 THEN P(R3)=P(R3)+INT(MA*15):: IF MA<Q THEN ER(R3,4)=-1 :: MAN=MAN-1
1890 CALL TI :: CALL DELSPRITE(#1):: CALL HCHAR(22,1,128,32)
1900 NEXT R3 :: NEXT R2 :: IF MAN=0 THEN 3000
2000 R1=R1+1 :: CALL TI :: IF R1>5 THEN R1=5 :: GOTO 100 ELSE 100
3000 CALL DELSPRITE(ALL):: CALL CLEAR :: R1=1 :: FOR I=1 TO AN :: IF P(I)<PL(8)T
HEN 3020 :: PL(8)=P(I):: P$(8)=SL$(I)
3010 FOR J=1 TO 8 :: FOR K=J+1 TO 8 :: IF PL(J)<PL(K)THEN A$=P$(J):: A=PL(J):: P
$(J)=P$(K):: PL(J)=PL(K):: P$(K)=A$ :: PL(K)=A
3015 NEXT K :: DISPLAY AT(3+2*J,7):STR$(J)&". "&P$(J):: DISPLAY AT(3+2*J,16)SIZE
(6):USING "######":PL(J):: NEXT J
3020 NEXT I
```

```
3025 CALL KEY(0,K,S):: IF S=0 THEN 3025 ELSE CALL CLEAR :: CALL CHARSET :: GOTO
40
10000 FOR I=1 TO 3 :: FO(I)=3 :: FOR J=1 TO 3 :: ER(I,J)=0 :: NEXT J :: NEXT I
10010 ON R GOSUB 10500,11000,11500,12000
10020 GOSUB 20000 :: RETURN
10500 A=.92 :: B=1.05
10510 DATA FFFFFFFFFFFFFFFF,,80C0A09088848281,,,A060202020202,80C0A0A0A090808,01
01,,2060A0202020202
10520 RESTORE 10510 :: FOR I=34 TO 43 :: READ A$ :: CALL CHAR(I,A$):: NEXT I
10530 CALL CHAR(59,"00E0A0E0000000000000003C4040403C000000446C5444440000003C4038
0478")
10540 DATA 00183C7E7E3C18,,,,,0001030709121404,000000000000040E,0F8EF8F070181404
10550 DATA 0001030303010303,0303030301030301,000080C08000808,8080080C08000008
10560 DATA 0001030303010708,07030323540B,000080C0800090A,C0808080402014O8
10570 DATA 0103030100010102,0408010101020505,0080C08080C0C0E,E0E0C0C0408,10383C3
9123C381F,1F0F06,0000800000207OC8,8502
10580 DATA FFFFFFFFFFFFFFFF,0080C0E0F0F8FCFE,FF7F3F1F0F070301,,,000000000020C818
,,,FFFFFFFFFFFFFFFF,1818181818181818,,
10590 DATA ,FF7F3F1F0F070301,,,,80C06030180C0602,,,183C5A7E24187EFF,,,,,00000000
00010301,,0000000000008
10600 FOR I=92 TO 143 :: READ A$ :: CALL CHAR(I,A$):: NEXT I
10610 DATA 12,11,10.5,10,9.5
10620 FOR I=1 TO R1 :: READ Q :: NEXT I :: RETURN
11000 A=.87 :: B=1.1 :: A$="vttttttttuuww" :: B$=CHR$(128)&"ttttttttuw" :: C$="w
wvttttttttu"
11010 DATA 6.5,7.25,7.75,8,8.25
11020 RESTORE 11010 :: FOR I=1 TO R1 :: READ Q :: NEXT I :: RETURN
11500 CALL CHAR(108,"406070584C464371"&RPT$("0",30)&"80C04040404040070")
11510 CALL CHAR(112,"0000000000FE8381"&RPT$("0",30)&"80C06030180CFE808")
11515 CALL CHAR(140,"0103030301071B0323570800000000000080C08030C08080A0D408")
11520 A=.91 :: B=1.05
11530 DATA 14,13.25,12.5,12,11.75
11540 RESTORE 11520 :: FOR I=1 TO R1 :: READ Q :: NEXT I :: RETURN
12000 DATA 36,,0101020204040808,1010202040408O8,,40,,01020408102,000004081020408
,,96,,030C30C,00000000030C30C,
12010 DATA 128,,FF,,FF,120,00000000C0300C03,,,C0300C03,132,0000201008040201,,,80
4020100804,140,0808040402020101,,,808040402020101
12020 RESTORE 12000 :: FOR I=1 TO 7 :: READ SP(I):: FOR J=0 TO 3 :: READ A$ :: C
ALL CHAR(SP(I)+J,A$):: NEXT J :: NEXT I :: A=.87 :: B=1.10
12030 CALL CHAR(117,"809CA2222222221CFF01",100,"0001030303010738",102,"000080C08
030C08")
12040 CALL CHAR(104,"0001030303010738030303025408000000000080C0800080808080808040202
0406")
12050 FOR I=1 TO 9 :: CALL CHARPAT(48+I,A$):: CALL CHAR(107+I,A$):: NEXT I
12060 DATA 75,80,84,87,90
12070 FOR I=1 TO R1 :: READ Q :: NEXT I :: A$=RPT$(CHR$(128),16):: FOR I=1 TO 9
:: A$=A$&RPT$("v",9)&CHR$(107+I)&"u" :: NEXT I :: A$=A$&RPT$("v",36):: RETURN
20000 CALL DELSPRITE(ALL):: CALL CLEAR :: FOR I=1 TO 14 :: CALL COLOR(I,1,1):: N
EXT I
20020 CALL HCHAR(1,1,34,192):: CALL HCHAR(7,1,136,256):: CALL HCHAR(15,1,136,96)
:: CALL HCHAR(18,1,119,224):: CALL HCHAR(22,1,128,32)
20030 FOR I=2 TO 15 :: CALL HCHAR(I,4,32,26):: NEXT I
20040 CALL HCHAR(1,3,124,28):: CALL HCHAR(7,3,124,28):: CALL HCHAR(16,3,124,28):
: CALL VCHAR(1,3,124,16):: CALL VCHAR(2,17,124,5)
20050 CALL VCHAR(1,30,124,16):: CALL VCHAR(11,19,125,4)
20060 DISPLAY AT(9,11)SIZE(7):"QUALIFY" :: DISPLAY AT(3,19)SIZE(6):"RECORD" :: D
ISPLAY AT(5,17)SIZE(3):WT$(R)
20070 FOR I=1 TO AN :: DISPLAY AT(2+I,3)SIZE(3):SL$(I):: DISPLAY AT(2+I,8)SIZE(6
):USING "######":P(I):: NEXT I
20080 FOR I=2 TO 10 :: CALL COLOR(I,16,1):: NEXT I :: CALL COLOR(1,8,1,11,12,4,1
2,14,1,13,12,9,14,11,5):: RETURN
21000 SUB WR(R,S$,W(),W$(),E)
21010 IF R=1 OR R=3 THEN IF E>W(R)THEN SUBEXIT ELSE 21030
21020 IF E<W(R)THEN SUBEXIT
21030 W(R)=E :: W$(R)=S$ :: CALL SOUND(1000,440,0,220,2,550,2):: FOR I=1 TO 12 :
: DISPLAY AT(3,19)SIZE(6):"  NEW " :: FOR J=1 TO 50 :: NEXT J
```

```
21040 DISPLAY AT(3,19)SIZE(6):"RECORD" :: FOR J=1 TO 50 :: NEXT J :: NEXT I :: D
ISPLAY AT(5,17):W$(R):: IF R=1 OR R=3 THEN DISPLAY AT(5,21)SIZE(6):USING 30:W(R)
:: GOTO 21060
21050 DISPLAY AT(5,21)SIZE(6):USING 31:W(R)
21060 CALL HCHAR(5,30,124):: SUBEND
22000 SUB Z(AN,E(,),S$()):: FOR I=1 TO AN :: IF E(I,4)THEN 22020
22010 DISPLAY AT(10+I,3)SIZE(3):S$(I):: FOR J=1 TO 3 :: CALL SPRITE(#32-3*I-J,92
,16,73+I*8,57+J*8):: NEXT J
22020 NEXT I :: SUBEND
25500 SUB Z1(Q,W,S$,R2,R3,MA,ER(,)):: MA1=0
25530 DISPLAY AT(9,20)SIZE(6):USING 31:Q :: DISPLAY AT(5,21)SIZE(6):USING 31:W
: DISPLAY AT(11,19)SIZE(3):S$ :: FOR I=1 TO R2
25540 CALL COLOR(#28-R3*3+I,9):: DISPLAY AT(11+I,18)SIZE(9):STR$(I)&"." :: NEXT
I :: FOR I=1 TO R2-1 :: IF ER(R3,I)=-1 THEN DISPLAY AT(11+I,21)SIZE(6):" FOUL" :
: MA1=-1 :: GOTO 25550
25545 DISPLAY AT(11+I,21)SIZE(6):USING 31:ER(R3,I):: MA=MAX(MA,ER(R3,I))
25550 NEXT I :: IF MA1=-1 AND MA=0 THEN DISPLAY AT(10+R3,11)SIZE(6):" FOUL" ELSE
 IF MA THEN DISPLAY AT(10+R3,11)SIZE(6):USING 31:MA
25560 SUBEND
26000 SUB TI :: FOR I=1 TO 1500 :: NEXT I :: SUBEND
28010 SUB Z2(M,E,R2,R3):: IF E<0 THEN E=0
28020 DISPLAY AT(11+R2,21)SIZE(6):USING 31:E :: M=MAX(M,E):: DISPLAY AT(10+R3,11
)SIZE(6):USING 31:M :: SUBEND
```

Universal Empire

für Schneider CPC 464

Erobern Sie das Universum

Wir befinden uns auf der geeinten Erde des Jahres 3415.
Seit der Entwicklung der Überlichtraumfahrt hat sich die terranische Industrie dermaßen explosiv entwickelt, daß man nun über modernstes Know-How verfügt. Politisch allerdings kann man solche Erfolge nicht vorweisen. Nach einhundertjährigen diplomatischen Verhandlungen ist es zu keinerlei Verständigung zwischen den Welten gekommen, die Lage spitzte sich sogar in den letzten Jahren immer mehr zu. Terra, einer der aufstrebendsten Planeten des bekannten Universums, wird nun von einem Angriff einiger protektionistischer, alt-dekadenter Planeten bedroht. Daher entschied der Terranische Föderationsrat, sofort eine schlagkräftige Flotte von 180 Einheiten bereitzustellen. Nach Scheitern aller friedlicher Missionen und der erheblichen Bedrohung von außen gibt man einem jungen Offizier der Flotte die Chance seines Lebens: Sie sollen ein universelles Imperium, das UNIVERSAL EMPIRE, mit der Erde als Zentrum schaffen. Dazu bekommen Sie den alleinigen Oberbefehl über die Truppe, einen genügenden Etat, 60 000 Tage Zeit und das Recht, auf allen Planeten des Imperiums Steuern einzuziehen.
Doch ganz so einfach wird die Aufgabe nicht. Einige Planeten haben Flotten von mehr als 10facher Größe, andere sind Ihnen an Intelligenz und Wissen überlegen. Allerdings können Sie hochentwickelte Welten zum Schiffbau einsetzen und auf einigen Welten auch fertige Schiffe kaufen. Das Geld dafür können Sie u. a. auch durch Handel mit allen Welten erhalten. Allerdings sind Sie auf fremden Welten stets in der Gefahr, angegriffen zu werden. Um eine noch stolzere Flotte zu erhalten, können Sie sich auch in Tiefschlaf versetzen. Ihr CPC 464 wird zum Monitor des Kommandoschiffes. Helfen Sie der Erde.
Gesteuert wird über eine ausgeklügelte Menütechnik. Mit den Tasten 0,1,2,3,., wird der Cursor über die Buchstaben bewegt bzw. wird die gewünschte Funktion ausgelöst. Falls Sie einen Joystick haben, können Sie Ihn für einige Funktionen benutzen.
Viel Vergnügen!

```
1 INK 2,20:INK 3,6
2 ENV 2,1,0,10,1,-10,0,1,0,10,1,-11,0
3 ENV 3,4,2,2,4,-2,2,4,2,2,4,-2,2
4 ENT 1,5,5,2,4,-2.5,2,3,-2.5,6,3,-2.5,10
5 ENV 1.5,2,3,4,5,3,2,10,3,2,20,3
6 'UNIVERSAL EMPIRE (c) 1984 by Marcus Schneider
7 SYMBOL AFTER 64
8 SYMBOL 65,24,36,66,126,98,98,98,0
9 SYMBOL 66,124,34,34,60,50,50,124,0
10 SYMBOL 67,28,34,64,96,96,34,28,0
11 SYMBOL 68,120,36,34,50,50,52,120,0
12 SYMBOL 69,126,64,64,126,96,96,126,0
13 SYMBOL 70,126,64,64,126,96,96,96,0
14 SYMBOL 71,28,34,64,110,98,34,28,0
```

Maschinensprache-Monitor für den TI-99

mit 32 k-Speichererweiterung

Will man in Maschinensprache programmieren, braucht man neben einem Assembler einen Maschinensprache-Monitor, mit dem man Speicherinhalte ausgeben, die Register anzeigen, Bytes suchen und Maschinenprogramme aufrufen lassen kann. Der folgende 99/4-Monitor bietet all dies und noch einiges mehr. Er läuft auf dem TI-99 mit Extended Basic und 32K-Speichererweiterung.

Der Monitor selbst ist in Extended Basic geschrieben, aber die wichtigsten Teile wie Speicherdumps und Suche nach Bytes sind in Maschinensprache; einerseits aus Geschwindigkeitsgründen und andererseits, weil viele Systemfunktionen in Basic gar nicht erreichbar sind.
Sobald Sie das Programmlisting abgetippt haben (achten Sie wegen der Prüfsummen auf die Zeilennummern!), können Sie das Programm starten. In Zeile 220 wird mit CALL CHECKSUM ein Unterprogramm aufgerufen, das die DATA-Zeilen auf ihre Richtigkeit prüft. Wird ein Fehler entdeckt, so stoppt das Programm und die Zeilennummer der fehlerhaften DATA-Zeile wird angegeben. Sind so alle DATAS geprüft, können Sie die Zeilen 220 und 2260 bis 2380 löschen und das Programm abspeichern. Ist die Diskette angeschlossen, Sie wollen das Programm aber auf Cassette abspeichern, kommt es wegen der Programmlänge zu Schwierigkeiten. Geben Sie in diesem Fall vor dem Abtippen CALL FILES(1) und NEW ein. Der Monitor selbst ist sehr einfach zu bedienen: Sie haben siebzehn Befehle zur Verfügung, die Sie aufrufen, indem Sie deren Anfangsbuchstaben eintippen. Drücken Sie z. B. „R" (ohne ENTER), werden die aktuellen Workspace-Register ausgegeben. Die folgende Tabelle zeigt alle Befehle mit Beschreibung:

A = „Assembler" – verknüpft das Programm mit einem Assembler, sofern ein solcher sich auf Cassette oder Diskette befindet (CS1 oder DSK1.ASSEMBLER).

B = „Breakpoint" – unterbricht den Monitor. Sie befinden sich wieder im Direktmodus und können Basic-Befehle verwenden. Der Monitor wird dann mit CONTINUE fortgesetzt.

D = „Decimal" – wandelt eine Dezimalzahl in eine Hexadezimal-Zahl um.
Beispiel:
>D 4096 = >1000

E = „Erase Screen" – löscht den Bildschirm.

F = „Find Word or Byte" – sucht ein Byte oder Word (2 Bytes) innerhalb vorgegebener Grenzen im Speicher – und zwar sowohl im normalen RAM/ROM als auch im VDP-RAM (Bildschirmspeicher) und im GROM. Dementsprechend können Sie der Startadresse ein „G" oder „V" voranstellen. Alle Fundstellen werden hexadezimal mit Adresse und Wert angegeben.
Beispiele:
>F WORD OR BYTE? W
START = 2000
END = 3FFF
VALUE = 02E0 sucht im Low Memory alle LWPI-Befehle (>02E0)
>F WORD OR BYTE? B
START = G0000
END = C000
VALUE = 00 sucht im GROM alle Null-Bytes

Mit ENTER können Sie die Suche beenden. Jede andere Taste führt dazu, daß das Listing angehalten wird; durch einen neuen Tastendruck geht es wieder weiter. Die Suche erfolgt übrigens in Maschinensprache mit der entsprechenden Geschwindigkeit.

G = „GROM Word Read" – liest ein Word aus dem GROM von der angegebenen Adresse. Mittels „+" und „–" können Sie den Speicher durchschreiten; mit ENTER können Sie dieses Unterprogramm verlassen.
Beispiel:
>G 0000 = AA02 –
FFFE = 0000 (Enter)

H = „Hexadezimal" – ist die Umkehrung des D-Befehls und wandelt eine Hexadezimalzahl in eine Dezimalzahl um:
>H A000 = -24576

I = „Iventory" – ist eine HELP-Funktion und listet auf dem Bildschirm alle Befehle auf.

J = „Jump" – ruft ein Maschinenprogramm mit seinem Namen auf, sofern dieser in der Namenstabelle steht:
>J ASMPRG

L = „Load" – lädt mittels CALL LOAD („Datei") ein Maschinenprogramm von der Diskette oder vom Cassettenrecorder:
>L DSK1.OBJECT 1

M = „Memory Read or Write" – funktioniert genauso wie das G-Kommando, nur daß Sie hier zusätzlich die Möglichkeit haben, an der Stelle, an der „+" oder „–" eingegeben wird, einen Hexwert einzutippen, der dann gepoked wird.
>M 24F4 = FFE0 +
24F6 = FE00 3FFF
24F8 = 03FF –
24F6 = 3FFF

Q = „Quit" – beendet die Arbeit mit dem Monitor, es erscheint die READY-Meldung.

R = „Registers Show" – holt sich den Workspace Pointer und zeigt daraufhin die sechzehn Workspace-Register an:
>R WP = 83 E0
R0=0001 R1=0010 R2=3F33
...
>S „Show Memory Dump" – ruft ein Maschinenprogramm auf, mit dem ein Memory-Dump von RAM, ROM, GROM und VDP-RAM angefertigt werden kann.
Beispiele:
>S 0000 00FF
>S G0000 01FF
>S V3FAC 3FFF
Die Tastenbedienung ist dieselbe wie bei der Find-Funktion.

T = „Text Fill" – ermöglicht es Ihnen, den RAM mit einem Text zu füllen:
>T 3000
DIES IST EIN PROBETEXT (Enter)

V = „VDP RAM Read/Write" – funktioniert genauso wie das M-Kommando, allerdings für den VDP-RAM:
>V 3FAC = 00FF +
3FAE = 3000 2000
3FB0 = FF35 (Enter)

X = „Execute Program" ruft ein Maschinensprache-Programm nicht mit den Namen auf, sondern mit der Startadresse und dem Workspace-Pointer:
>X
WP = 83E0 (Workspace-Pointer)
PC = 24FC (Programm-Zähler, Programm-Counter)
ACHTUNG: Steht an der angegebenen Adresse kein ausführbares Programm, so steigt der Computer aus und kann dann nur mit Reset zurückgeholt werden. Der Monitor muß dann natürlich neu geladen werden.

Die meisten Kommandos können Sie auch vorzeitig abbrechen, wenn Sie statt eines Wertes einfach „Enter" eingeben. Folgende Maschinenprogramme sind implementiert, die natürlich auch anderswo verwendet werden können:
- CALL LINK(„GETWP",X,WP) liefert den Workspace Pointer in WP.
- CALL LINK(„EXEC",WP,PC) ruft ein Maschinenprogramm nach der Adresse auf.
- CALL LINK(„MDUMP",Start,Ende) gibt auf dem Screen einen Memory Dump aus.
- CALL LINK(„VDUMP",Start,Ende) macht dasselbe für den VDP-RAM.
- CALL LINK(„GDUMP",Start,Ende) auch wieder ein Screen Dump, jedoch im GROM.
- CALL LINK(„VDEEK",Adr,Var) liest zwei Bytes (ein Word) aus dem VDP-RAM.
- CALL LINK(„MDEEK",Adr,Var) entspricht CALL PEEK(Adr,A,B) und Var=A*256+B.
- CALL LINK(„GDEEK",Adr,Var) macht dasselbe für den GROM.
- CALL LINK(„VDOKE",Adr,Wert) poked zwei Bytes in den VDP-RAM.
- CALL LINK(„MDOKE",Adr,Wert) poket zwei Bytes in den normalen RAM.
- CALL LINK(„MFIND",Start, Ende,-Wert,WB) sucht den RAM oder ROM nach einem Byte (WB=1) oder einem Word (WB=0) ab.
- CALL LINK(„GFIND",Start, Ende-,Wert,WB) und
- CALL LINK(VFIND",Start,Ende-,Wert,WB) tun dasselbe im GROM oder VDP-RAM.

Das Maschinenprogramm liegt übrigens zusammen mit dem Basic-Programm im High Memory ab –18912. Somit bleibt der gesamte Low Memory für Maschinenprogramme frei. Sie können natürlich auch absoluten Maschinencode ab >A000 laden, müssen aber darauf aufpassen, daß er dann nicht mit dem Monitor zusammenstößt.

Martin Kotulla

```
100 !  ********************
110 ! **********************
120 ! *                    *
130 ! *   99/4-MONITOR     *
140 ! *                    *
150 ! **********************
160 !  ********************
170 !
180 ! ----------------------
190 ! (C) Martin Kotulla,
200 ! ----------------------
210 !
220 CALL CHECKSUM ! Kann nach Probelauf zusammen mit 2260-2380 geloescht werden
230 CALL ASMPROG :: ON WARNING NEXT :: HX$="0123456789ABCDEF" :: ON ERROR 1410
240 DEF HEX$(X)=SEG$(HX$,INT(X/16)+1,1)&SEG$(HX$,X-INT(X/16)*16+1,1)
250 DEF HEXW$(X)=HEX$(INT(X/256))&HEX$(X-INT(X/256)*256)
260 DEF DEZ(X$)=(POS(HX$,SEG$(X$,1,1),1)-1)*16+POS(HX$,SEG$(X$,2,1),1)-1
270 DEF DEZW(X$)=DEZ(SEG$(X$,1,2))*256+DEZ(SEG$(X$,3,2))
280 DEF MS(X)=INT(X/256):: DEF LS(X)=X-INT(X/256)*256
290 DEF FU(X)=INT(X/2)*2
295 DEF F$(X$)=RPT$("0",4-LEN(X$))&X$
300 DEF PO(X)=X-65536*(X<0):: DEF NE(X)=X+65536*(X>32767)
310 DEF HPO$(X)=HEXW$(PO(X)):: DEF DNEG(X$)=NE(DEZW(F$(X$)))
320 DISPLAY AT(21,6)ERASE ALL:"** 99/4 MONITOR **" :: GOTO 350
330 PRINT
340 PRINT
350 CALL HCHAR(24,2,62)
360 CALL KEY(0,K,S):: IF K=13 OR K=32 THEN 340
370 IF S=0 OR K<65 OR K>90 THEN 360 ELSE K$=CHR$(K):: PRINT K$
380 ! ## A ##
390 IF K$<>"A" THEN 450
400 DISPLAY AT(23,5):"CS1 OR DSK1?"
410 ACCEPT AT(23,20)SIZE(-1):NA$
420 IF NA$="C" THEN RUN "CS1"
430 IF NA$="D" THEN RUN "DSK1.ASSEMBLER" ELSE 340
440 ! ## D ##
450 IF K$<>"D" THEN 490
```

```
460 ACCEPT AT(23,3)SIZE(6)VALIDATE(DIGIT,"-"):A
470 DISPLAY AT(23,14):"=  )";HPO$(A):: GOTO 340
480 ! ## E ##
490 IF K$="E" THEN 320
500 ! ## H ##
510 IF K$()"H" THEN 550
520 ACCEPT AT(23,3)SIZE(4)VALIDATE(HX$):A$
530 DISPLAY AT(23,14):USING "= ######":DNEG(A$):: GOTO 340
540 ! ## M/G/V ##
550 IF K$="M" OR K$="G" OR K$="V" THEN M$=K$ ELSE 660
560 ACCEPT AT(23,3)VALIDATE(HX$)SIZE(4):A$ :: A$=F$(A$)
570 ADR=DNEG(A$):: A$=HPO$(ADR):: PRINT
580 CALL LINK(M$&"DEEK",ADR,X):: X=PO(X)
590 PRINT A$;"=";HEXW$(X);"  ";CHR$(MS(X)AND 127);CHR$(LS(X)AND 127)
600 ACCEPT AT(23,17)SIZE(4)VALIDATE(HX$,"+-"):A$
610 IF A$="" THEN 340
620 IF A$="+" THEN ADR=ADR+2 ELSE IF A$="-" THEN ADR=ADR-2 ELSE 640
630 AD2=PO(ADR):: ADR=NE(AD2):: A$=HEXW$(AD2):: GOTO 580
640 CALL LINK(K$&"DOKE",ADR,DNEG(A$)):: ADR=ADR+2 :: GOTO 630
650 ! ## Q ##
660 IF K$="Q" THEN END
670 ! ## R ##
680 IF K$()"R" THEN 760 ELSE PRINT
690 RG=0 :: CALL LINK("GETWP",X,C):: DISPLAY AT(22,9):"WP=";HPO$(C)
700 FOR I=C TO C+30 STEP 2 :: CALL PEEK(I,A,B)
710 RG$=CHR$(RG+48-7*(RG)9))
720 PRINT "R";RG$;"=";HEX$(A);HEX$(B);" ";
730 RG=RG+1 :: IF INT(RG/3)=RG/3 THEN PRINT
740 NEXT I :: GOTO 330
750 ! ## J ##
760 IF K$()"J" THEN 810
770 ACCEPT AT(23,3)SIZE(6):A$
780 IF A$()"" THEN PRINT :: CALL LINK(A$)
790 GOTO 340
800 ! ## L ##
810 IF K$()"L" THEN 840
820 ACCEPT AT(23,3):A$ :: CALL LOAD(A$):: GOTO 340
830 ! ## T ##
840 IF K$()"T" THEN 890
850 ACCEPT AT(23,3)VALIDATE(HX$):A$ :: ADR=DNEG(A$):: PRINT : :
860 CALL KEY(0,K,S):: IF S(1 THEN 860 ELSE IF K=13 THEN 330
870 PRINT CHR$(K);:: CALL LOAD(ADR,K):: ADR=NE(ADR+1):: GOTO 860
880 ! ## X ##
890 IF K$()"X" THEN 940
900 PRINT :"WP=" :: ACCEPT AT(23,4):WP$ :: IF WP$="" THEN 340
910 PRINT "PC=" :: ACCEPT AT(23,4):PC$ :: IF PC$="" THEN 340
920 CALL LINK("EXEC",DNEG(WP$),DNEG(PC$)):: GOTO 340
930 ! ## S ##
940 IF K$()"S" THEN 1030
950 ACCEPT AT(23,3)SIZE(5)VALIDATE(HX$,"GV"):A$
960 IF A$="" THEN 340 ELSE AS=ASC(A$)
970 IF AS=86 OR AS=71 THEN NA$=CHR$(AS):: A$=SEG$(A$,2,4)ELSE NA$="M"
980 IF A$="" THEN 340 ELSE ST=DNEG(SEG$(A$,LEN(A$)-3,4))
990 ACCEPT AT(23,9)SIZE(4)VALIDATE(HX$):A$
1000 IF A$="" THEN 340 ELSE EN=DNEG(A$)
1010 PRINT :: CALL LINK(NA$&"DUMP",ST,EN):: GOTO 340
1020 ! ## F ##
1030 IF K$()"F" THEN 1180
1040 DISPLAY AT(23,3):"WORD OR BYTE?"
1050 ACCEPT AT(23,17)SIZE(1)VALIDATE("WB"):WB$
1060 IF WB$="B" THEN SI=2 :: WB=1 :: GOTO 1080
1070 IF WB$="W" THEN SI=4 :: WB=0 ELSE 340
1080 PRINT :"START =" :: ACCEPT AT(23,9)SIZE(5)VALIDATE(HX$,"GV"):ST$
1090 IF ST$="" THEN 340 ELSE AS=ASC(ST$)
1100 IF AS=86 OR AS=71 THEN NA$=CHR$(AS):: ST$=SEG$(ST$,2,4)ELSE NA$="M"
1110 ST=DNEG(SEG$(ST$,LEN(F$(ST$))-3,4))
```

```
1120 PRINT "END   =" :: ACCEPT AT(23,9)SIZE(4)VALIDATE(HX$):EN$
1130 IF EN$="" THEN 340 ELSE EN=DNEG(EN$)
1140 PRINT "VALUE =" :: ACCEPT AT(23,9)SIZE(SI)VALIDATE(HX$):VA$
1150 IF VA$="" THEN 340 ELSE VA=DNEG(VA$)
1160 PRINT :: CALL LINK(NA$&"FIND",ST,EN,VA,WB):: GOTO 340
1170 ! ## I ##
1180 IF K$()"I" THEN 1370 ELSE PRINT
1190 PRINT "A = RUN ASSEMBLER"
1200 PRINT "B = BREAKPOINT IN MONITOR"
1210 PRINT "D = DECIMAL TO HEXADECIMAL"
1220 PRINT "E = ERASE SCREEN"
1230 PRINT "F = FIND WORD OR BYTE"
1240 PRINT "G = READ WORD FROM GROM"
1250 PRINT "H = HEXADECIMAL TO DECIMAL"
1260 PRINT "I = INVENTORY OF COMMANDS"
1270 PRINT "J = JUMP INTO PROGRAM"
1280 PRINT "L = LOAD PROGRAM FROM DISK"
1290 PRINT "M = MEMORY READ OR WRITE"
1300 PRINT "Q = QUIT MONITOR"
1310 PRINT "R = SHOW WORKSPACE REGISTERS"
1320 PRINT "S = SHOW MEMORY DUMP"
1330 PRINT "T = FILL MEMORY WITH TEXT"
1340 PRINT "V = VDP RAM READ OR WRITE"
1350 PRINT "X = EXECUTE PROGRAM" :: GOTO 340
1360 ! ## B ##
1370 IF K$()"B" THEN 1390 ELSE BREAK
1380 GOTO 340
1390 CALL SOUND(150,220,0):: PRINT :"* ILLEGAL COMMAND" :: GOTO 340
1400 ! ********************
1410 CALL ERR(EC,ET,ES,EL):: ON ERROR 1410 :: CALL SOUND(150,220,0)
1420 IF EC=130 THEN PRINT "* I/O ERROR" :: RETURN 340
1430 IF EC=135 THEN PRINT "* SUBPROGRAM NOT FOUND" :: RETURN 340
1440 IF EC=74 THEN PRINT "* INCORRECT ARGUMENT LIST" :: RETURN 340
1450 PRINT :"* ERROR #";STR$(EC);" IN";EL :: CALL LOAD(-31932,0)
1460 ! ********************
1470 SUB ASMPROG :: RESTORE
1480 DISPLAY AT(8,9)BEEP ERASE ALL:"99/4-MONITOR"
1490 CALL HCHAR(10,3,42,28)
1500 DISPLAY AT(15,9):"BITTE WARTEN": :TAB(8);"ICH LESE DATEN"
1510 DISPLAY AT(24,3):"(C) MARTIN KOTULLA 1984"
1520 CALL PEEK(8198,A):: IF A()170 THEN CALL INIT
1530 FOR I=-18912 TO -18012 STEP 10 :: READ A,B,C,D,E,F,G,H,J,K
1540 CALL LOAD(I,A,B,C,D,E,F,G,H,J,K):: NEXT I
1550 CALL PEEK(8196,A,B):: C=A*256+B-112
1560 CALL LOAD(8196,INT(C/256),C-INT(C/256)*256)
1570 RESTORE 2180
1580 FOR I=C TO C+112 STEP 8 :: READ A,B,C,D,E,F,G,H
1590 CALL LOAD(I,A,B,C,D,E,F,G,H):: NEXT I
1600 !
1610 DATA 048,049,050,051,052,053,054,055,056,057,065,066,067,068,069,070
1620 DATA 032,000,013,000,127,030,032,032,032,032,061,032,032,032,032,032
1630 DATA 032,032,032,032,032,032,032,032,032,032,032,032,032,032,032,032
1640 DATA 032,032,016,000,010,142,000,000,000,000,002,160,200,000,131,074
1650 DATA 002,224,182,000,006,160,185,122,004,096,185,150,002,224,182,000
1660 DATA 002,001,000,001,006,160,185,106,200,032,131,074,182,132,006,160
1670 DATA 185,106,002,224,131,224,192,032,131,074,006,144,004,096,185,150
1680 DATA 002,224,182,000,006,160,185,076,002,001,000,001,006,160,185,106
1690 DATA 192,096,131,074,006,160,185,092,216,032,152,000,131,074,216,032
1700 DATA 152,000,131,075,006,160,185,122,004,096,185,140,004,096,185,150
1710 DATA 002,224,182,000,002,001,000,001,006,160,185,106,192,224,131,074
1720 DATA 006,160,035,202,200,001,131,074,006,160,185,122,004,096,185,150
1730 DATA 002,224,182,000,002,001,000,001,006,160,185,106,193,032,131,074
1740 DATA 006,160,185,106,192,096,131,074,006,160,035,230,004,096,185,150
1750 DATA 002,224,182,000,002,001,000,001,006,160,185,106,192,160,131,074
```

```
1760 DATA 200,018,131,074,006,160,185,122,004,096,185,150,002,224,182,000
1770 DATA 002,001,000,001,006,160,185,106,193,032,131,074,006,160,185,106
1780 DATA 197,032,131,074,004,096,185,150,002,224,182,000,002,012,184,212
1790 DATA 016,009,002,224,182,000,002,012,184,238,016,004,002,224,182,000
1800 DATA 002,012,184,216,002,001,000,004,006,160,185,106,193,096,131,074
1810 DATA 019,006,200,032,182,084,183,152,007,032,182,086,016,005,200,032
1820 DATA 182,082,183,152,004,224,182,086,006,160,185,076,002,001,000,001
1830 DATA 006,160,185,106,193,096,131,074,006,160,185,106,193,160,131,074
1840 DATA 006,160,185,106,195,160,131,074,016,000,200,032,182,086,182,086
1850 DATA 022,021,006,156,130,078,019,004,129,133,018,251,004,096,185,140
1860 DATA 194,069,006,073,002,004,182,054,006,160,184,166,194,078,006,160
1870 DATA 184,166,002,003,000,009,006,160,185,000,016,235,006,156,004,224
1880 DATA 182,088,147,137,019,011,006,201,147,137,022,004,005,133,007,032
1890 DATA 182,088,016,004,129,133,018,242,004,096,185,140,194,069,006,073
1900 DATA 002,004,182,054,006,160,184,166,194,078,006,160,184,166,002,003
1910 DATA 000,007,006,160,185,000,200,032,182,088,182,088,019,223,006,005
1920 DATA 016,221,002,224,182,000,002,012,184,212,016,009,002,224,182,000
1930 DATA 002,012,184,238,016,004,002,224,182,000,002,012,184,216,006,160
1940 DATA 185,076,002,001,000,001,006,160,185,106,193,096,131,074,006,160
1950 DATA 185,106,193,160,131,074,002,004,182,054,194,069,006,160,184,166
1960 DATA 006,160,184,116,200,009,182,076,006,160,184,116,200,009,182,078
1970 DATA 006,160,184,116,200,009,182,080,006,160,184,252,129,133,018,235
1980 DATA 004,096,185,140,194,139,006,156,006,160,184,166,002,073,127,127
1990 DATA 152,009,182,048,017,003,152,009,182,052,017,002,210,096,182,053
2000 DATA 006,201,152,009,182,048,017,003,152,009,182,052,017,002,210,096
2010 DATA 182,053,006,201,004,090,192,137,009,194,221,034,182,032,192,137
2020 DATA 002,066,015,000,009,130,221,034,182,032,192,137,002,066,000,240
2030 DATA 009,066,221,034,182,032,192,137,002,066,000,015,221,034,182,032
2040 DATA 005,132,004,091,194,117,004,091,195,075,192,069,006,160,185,092
2050 DATA 210,096,152,000,216,032,152,000,182,019,005,197,004,093,195,075
2060 DATA 192,197,006,160,035,202,194,065,005,197,004,093,002,003,000,028
2070 DATA 002,000,002,226,002,002,182,054,208,114,002,033,096,000,004,032
2080 DATA 032,032,005,128,006,003,022,248,004,032,032,024,000,038,004,032
2090 DATA 032,028,193,224,131,124,033,224,182,048,022,015,152,032,131,117
2100 DATA 182,050,019,044,004,032,032,028,193,224,131,124,033,224,182,048
2110 DATA 022,249,152,032,131,117,182,050,019,033,004,091,210,032,152,002
2120 DATA 006,200,210,032,152,002,006,200,006,008,004,091,216,001,156,002
2130 DATA 006,193,216,001,156,002,006,193,004,091,004,192,004,032,032,012
2140 DATA 004,032,032,024,018,184,005,129,004,091,004,192,002,001,000,002
2150 DATA 004,032,032,024,000,032,004,032,032,008,004,091,216,008,156,002
2160 DATA 006,200,216,008,156,002,216,032,000,002,131,124,002,224,131,224
2170 DATA 004,096,000,112,000,001,002,003,004,000,000,000,000,000,000,000
2180 DATA 071,070,073,078,068,032,183,076,086,070,073,078,068,032,183,066
2190 DATA 077,070,073,078,068,032,183,056,071,068,079,075,069,032,182,188
2200 DATA 077,068,079,075,069,032,183,028,086,068,079,075,069,032,182,224
2210 DATA 071,068,069,069,075,032,182,144,077,068,069,069,075,032,183,000
2220 DATA 086,068,069,069,075,032,182,192,071,068,085,077,080,032,184,038
2230 DATA 086,068,085,077,080,032,184,028,077,068,085,077,080,032,184,018
2240 DATA 069,088,069,067,032,032,182,108,071,069,084,087,080,032,182,090
2250 SUBEND
2260 ! **********************
2270 SUB CHECKSUM :: RESTORE 2330 :: DIM CHK(64)
2280 PRINT "BITTE WARTEN! ICH PRUEFE":"DIE DATA-ZEILEN!": :
2290 FOR I=1 TO 64 :: READ CHK(I):: NEXT I :: RESTORE 1610
2300 FOR I=1 TO 64 :: SUM=0 :: FOR J=1 TO 8 :: READ A,B :: SUM=SUM+A+B :: NEXT J
2310 IF SUM=CHK(I)THEN DISPLAY BEEP:I*10+1530;"KORREKT!" ELSE CALL SOUND(150,220
,0):: PRINT " FEHLER IN";I*10+1530 :: STOP
2320 NEXT I :: PRINT "DIE DATA'S SIND KORREKT!"
2330 DATA 930,551,512,799,1724,1378,1886,1296,1789,1691,1490,1717,1299,1816,1426
2340 DATA 1739,1348,1687,1297,1372,1418,1526,1966,1801,1392,1538,1356,1278,1579
2350 DATA 1551,1383,1488,1444,1382,1870,1868,2054,1765,1366,1338,1866,1328,1425
2360 DATA 1612,1628,1383,959,642,1688,1550,1478,1292,1144,724,677,1674,222,1307
2370 DATA 1401,1426,1283,1408,1261,1342
2380 SUBEND
```

Hardcopy für TI-99

Beginn des Hardcopyunterprogramms: Dimensionierung der sogenannten binären Überlaufstrings für 32 Spalten. Kanal #1 wird auf hochauflösende Grafik eingestellt.
Binärstring (Zahlen von 0 bis 16 in Binärform aneinandergereiht) wird definiert.
Hexadezimalstring (Zahlen von 0 bis 16 in Hexadezimalform aneinandergereiht) wird definiert.
String aus 64 Nullen und ein Zeilenvorschub für hochauflösende Grafik werden definiert.
Bildschirmzeilenabtastschleife (von 1 bis 24).
Bildschirmspaltenabtastschleife (von 1 bis 32).
Der ASCII-Code einer der 768 Bildschirmpositionen wird ermittelt und der binäre Zeichencodestring wird gelöscht.
Falls der ASCII-Code nicht zwischen 32 und 143 liegt, ist der binäre Zeichencodestring gleich dem 64-Nullenstring und es geht in Zeile 30030 weiter.
Der hexadez. Zeichencodestring vom ermittelten ASCII-Code wird abgefragt. Falls der hexadez. Zeichencodestring aus 16 Nullen besteht, ist der bin. Zeichencodestring gleich dem 64-Nullenstring und es geht weiter in Zeile 30030.
Der 16stellige, hexadez. Zeichencodestring wird in einen 64stelligen, binären Zeichencodestring transformiert.
Ein bin. Überlaufstring wird mit dem bin. Zeichencodestring verknüpft. Das Ganze ist jetzt der neue bin. Zeichencodestring.
Der Teil des bin. Zeichencodestrings, der nach der 56. Position beginnt, wird ermittelt. Dieser Teil ist ein neuer bin. Überlaufstring.
Die ersten 56 Positionen des bin. Zeichencodestrings werden an das Druckunterprogramm übergeben.
Die Spaltenabtastschleife wird um eins erhöht, d. h. das nächste Bildschirmzeichen wird umgewandelt.
Zeilenvorschub auf dem Drucker.
Falls der bin. Überlaufstring der 32. Spalte 56 Positionen enthält, wird er als ganze Zeile auf dem Drucker ausgegeben und gelöscht, anschließend ein Zeilenvorschub, ansonsten erfolgt ein Sprung in Zeile 30050.
Die Zeilenabtastschleife wird um eins erhöht, d. h. die nächste Zeile auf dem Bildschirm wird nun spaltenweise abgetastet und transformiert. Den letzten bin. Überlaufstrings, die nicht aus 56 Positionen bestehen, werden einige Nullen hinzugefügt. Dann werden sie als ganze Zeile einzeln dem Druckunterprogramm übergeben.
Der letzte Zeilenvorschub auf dem Drucker.
Kanal #1 wird geschlossen, die Hardcopy ist erstellt.
Die langen Strings werden gelöscht.
Das Hardcopyprogramm ist beendet.
Beginn des Druckunterprogramms, der Parameter ist entweder der 56 Positionen beinhaltende bin. Zeichencodestring oder der bin. Überlaufstring. Matrixspaltenschleife für ein Zeichen, das aus einer 8 x 7 Matrix besteht, wird begonnen (von 1 bis 8). Die Variable für hochauflösende Druckergrafik wird auf Leerdruck zurückgestellt.
Matrixzeilenschleife (von 0 bis 6).
Zur Druckvariablen wird ein umgewandelter bin. Übergabestring addiert.
Nächste Matrixzeile.
Eine Spalte der Matrix wird an den Drucker übergeben.
Die nächste Matrixspalte wird berechnet.
Ende des Druckunterprogramms.

```
20000 ! ####################
20005 ! #                  #
20010 ! #                  #
20015 ! #                  #
20020 ! #  BASIC-HARDCOPY  #
20030 ! #   LONG-VERSION   #
20040 ! #   (C)BY KABOSO   #
20050 ! # KARSTEN  BOGUSCH #
20060 ! #     SOFTWARE     #
20070 ! #                  #
20080 ! #                  #
20090 ! #                  #
20100 ! ####################
20110 !
20120 !
20130 !
30000 SUB HARD
30001 DIM U$(32)
30002 OPEN #1:"RS232.DA=8.CR.LF" :: PRINT #1:CHR$(8);
30003 B$="0000000100100011010001010110011110001001101010111100110111101111"
30004 H$="0123456789ABCDEF"
30005 T$=RPT$("0",64):: C$=CHR$(13)&CHR$(10)
30010 FOR Z=1 TO 24
30011 FOR S=1 TO 32
30012 CALL GCHAR(Z,S,Q):: Z$=""
30013 IF Q<32 OR Q>143 THEN Z$=T$ :: GOTO 30030
```

```
30014 CALL CHARPAT(Q,Q$)
30015 IF Q$=SEG$(T$,1,16)THEN Z$=T$ :: GOTO 30030
30020 FOR I=1 TO 16
30021 Z$=Z$&SEG$(B$,1+4*(POS(H$,SEG$(Q$,I,1),1)-1),4)
30022 NEXT I
30030 Z$=U$(S)&Z$
30031 U$(S)=SEG$(Z$,57,LEN(Z$)-56)
30032 CALL D(SEG$(Z$,1,56))
30033 NEXT S
30034 PRINT #1:C$
30035 IF LEN(U$(32))<>56 THEN 30050
30040 FOR I=1 TO 32 :: CALL D(U$(I)):: U$(I)="" :: NEXT I
30041 PRINT #1:C$;
30050 NEXT Z
30051 FOR I=1 TO 32 :: CALL D(U$(I)&SEG$(T$,1,32)):: U$(I)="" :: NEXT I
30052 PRINT #1:C$;
30053 CLOSE #1
30054 B$,H$,T$=""
30055 SUBEND
30070 SUB D(G$)
30071 FOR N=1 TO 8 :: K=128
30072 FOR M=0 TO 6
30073 K=K+2^M*-(VAL(SEG$(G$,8*M+N,1))>0)
30074 NEXT M
30075 PRINT #1:CHR$(K);
30076 NEXT N
30077 SUBEND
```

Die geschichtliche Entwicklung der Datenverarbeitung im Zeitraffer

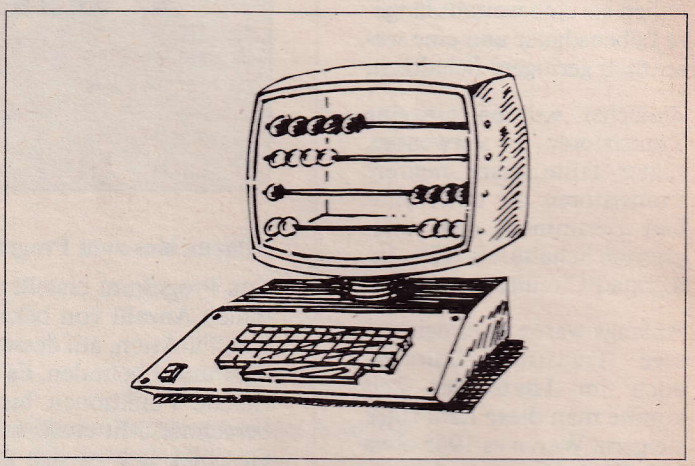
Von Oliver Steinmeier und Klaus Schneider

DIE ANFÄNGE DER DATENVERARBEITUNG UND MECHANISCHEN RECHENMASCHINEN

Schon lange vor den ersten Computern nutzten die Menschen Maschinen als Hilfsmittel zum Rechnen.
So wurde vor ca. 4000 Jahren der Abakus erfunden. Eingesetzt wurde er in Japan und China, aber auch die alten Römer haben ihn benutzt.
Auch heute noch erreichen die Japaner mit dem Abakus sehr hohe Rechengeschwindigkeiten. Experten sind in der Lage, 50 Divisionsaufgaben mit 5 bis 7stelligen Zahlen in etwa 90 Sekunden zu lösen. Sie addieren 15 zehnstellige Zahlen in nur 20 Sekunden.
Weitere Schritte zum modernen Rechner machte man ab dem 17. Jahrhundert.
Schon 1620 fiel dem englischen Mathematiker GUNTER auf, daß sich mit Logarithmen schneller rechnen läßt, wenn man auf einem Maßstab die logarithmischen Werte als Strecken vom Ausgangspunkt aufträgt. Die Genauigkeit ist dabei vom Maßstab abhängig. Der Rechenstab ist somit eine für jede Zahl einstellbare Logarithmentafel.
1623/24 schrieb WILHELM SCHICKARD eine Arbeit über Rechenmaschinen. Eine 1950 nach seinen Aufzeichnungen gebaute Vierspeziesrechenmaschine ist voll funktionsfähig. Sie ist heute im Deutschen Museum in München ausgestellt.
BLAISE PASCAL ließ im Jahre 1641 eine Additions- und Subtraktionsmaschine bauen, um Steuern leichter und schneller berechnen zu können.
Da SCHICKARDS Rechenmaschine in den Wirren des Dreißigjährigen Krieges verloren ging, galt PASCALS Maschine, obwohl sie primitiver war, lange Zeit als erste auf ihrem Gebiet.
Eine Maschine ähnlich der Pascals erfand G. W. LEIBNIZ. Dieses Gerät war jedoch einfacher konstruiert, da er das Prinzip der Staffelwalzen erfand. Außerdem erkannte Leibniz schon die Möglichkeiten der Dualzahlen in der Datenverarbeitung.
Das Lochkartenprinzip wurde in der Wirtschaft zuerst von J. M. JACQUARD benutzt. Er vervollständigte 1805 den mechanischen Musterwebstuhl durch Programmierung mit Lochkarten.
CHARLES BABBAGE entwickelte 1823 ebenfalls auf der Basis von Lochkarten eine Differenziermaschine, die zur Berechnung von Funktionen diente. Trotz großer Fortschrittlichkeit hatte er nur geringen Erfolg, da die Mechanik zu aufwendig war.
GEORGE BOOLE – nach ihm wurde die Boolesche Algebra benannt – steuerte die wichtigsten Grundlagen zur logischen Algebra bei. (0 für falsch, 1 für richtig).
HERMANN HOLLERITH (deutscher Vater) setzte bei der amerikanischen Volkszählung 1890 erstmals Lochkarten als Informationsträger im großen Stil ein. Dieses Prinzip fand auf Grund der großen Zeitersparnis auch in Österreich und Rußland Verbreitung. Vor der Lochkartenverarbeitung benötigten 500 Mitarbeiter 7 Jahre zur Auswertung der im Abstand von 5 Jahren stattfindenden Zählung, nun schaffte man diese Arbeit in etwa 7 Monaten und brauchte nur noch 43 Arbeitskräfte.

RECHNER MIT RELAISTECHNIK

1941 konstruierte der deutsche Forscher und Techniker KONRAD ZUSE den bekannten „Rechner" Z 3. Seine Maschine bestand aus ca. 2600 Relais; gesteuert wurde sie durch gelochten Kinofilmstreifen. Z 3 entwickelte



eine Rechengeschwindigkeit von 15 – 20 arithmetischen Operationen pro Sekunde. Eine Multiplikation dauerte etwa 4 – 5 Sekunden.
Z 3 konnte 64 Zahlen mit je 22 Dualstellen speichern und verarbeiten. Für den Speicher benötigte ZUSE 2000 Relais, 600 wurden im Rechenwerk eingesetzt.
Neben den vier Grundrechenarten waren das Wurzelziehen und die Multiplikation mit 2, ½, 10, $^1/_{10}$, -1 eingebaut. Zuse Z 3 kostete etwa 25 000 Reichsmark.

In den USA wurde 1944 MARK 1 durch HOWARD H. AIKEN mit Unterstützung von IBM an der Haward University in Cambridge/Massachusetts fertiggestellt. MARK 1 bestand aus 13 000 Relais und kostete damals 400 000 Dollar. Er wurde durch die Programmierung der Einzelmaschinen mittels Stecktafeln gesteuert.
Dieser Riesencomputer hatte eine Länge von 16 Metern und wog 35 Tonnen. Für eine Multiplikation benötigte er etwa 6 Sekunden.
AIKEN baute MARK 2, 3 und 4, bei denen erstmals Magnettrommeln und Magnetbänder als Speichermedium eingesetzt wurden.

COMPUTER MIT RÖHRENTECHNIK

In den Jahren 1943 – 46 wurde von Mitarbeitern der Moore-School of Electrical Engeneering der legendäre ENIAC (Elektronic Numerical Integrator and Computer) gebaut.
Er bestand aus 18 000 Elektronenröhren, die zusammen etwa 200 kW verbrauchten.
ENIAC stand auf einer Fläche von 135 Quadratmetern und kostete 10 Mio. Dollar.
Da Elektronenröhren eine geringe Lebensdauer haben, mußte bei 18 000 Röhren fast ständig nach Fehlern und fehlerhaften Röhren gesucht werden.
ENIAC war der erste Elektronenröhrenrechner und benötigte zur Multiplikation zweier 10stelliger Dezimalzahlen nur 2,8 Millisekunden.
Eingesetzt wurde er vorwiegend für ballistische Berechnungen.
Erst 1955 wurde er außer Dienst gestellt und steht heute im Nationalmuseum in Washington.

TRANSISTORTECHNIK IN DER DATENVERARBEITUNG

Nachdem 1954 der erste Silizium-Transistor gebaut wurde, konnte man nun die Computer statt der Elektronenröhren mit Transistoren bauen.

Transistoren haben gegenüber den Röhren viele Vorteile. Sie sind viel kleiner und haben eine geringere Stromaufnahme. Außerdem haben sie eine weitaus längere Lebensdauer und eine wesentlich geringere Schaltzeit.

Zunächst wurden einzelne Transistoren verwendet, später faßte man mehrere Transistoren zu einer Einheit zusammen, dem integrierten Schaltkreis (IC – Integrated Circuit).

Anfangs waren es einige wenige Transistorfunktionen, doch im Laufe der Zeit konnte man diese Zahl stark steigern. Waren es 1965 etwa 100 Funktionen, so konnte man 1970 schon 1000 Bauteile zusammenfassen (heute sind es mehrere 100 000).

CHIPS IN DER DATENVERARBEITUNG

1970 kam es zur Entwicklung des ersten Ein-Chip-Mikroprozessors durch den Amerikaner GARY BOONE von Texas Instruments.

Heute gibt es eine Vielzahl verschiedener Prozessoren mit unterschiedlichen Bus-Breiten (8, 16, 32 Bit). In den heutigen Home- und Personalcomputern werden hauptsächlich folgende Mikro-Prozessoren verwendet: 6502, 6809, Z-80A, 8088, 68000.

Es ist fast selbstverständlich, daß Chips heute mit Hilfe von Computern entwickelt und produziert werden.
Die Einsatzmöglichkeiten von Mikroprozessoren sind vielfältig. Sie rechnen und vergleichen, steuern und regeln, treffen logische Entscheidungen, sammeln und verarbeiten Daten.

So erfolgt die Abarbeitung des Waschprogramms einer Waschmaschine mit Hilfe eines solchen Mikroprozessors.

Die Entwicklung ist in diesem Bereich keineswegs abgeschlossen, es werden immer leistungsfähigere und schnellere Computer und Mikroprozessoren gebaut.

Funktionsanalyse für TI-99

Allgemeines zum Programm

Das Programm erstellt aus einer minimalen Anzahl von bekannten Punkten eine Funktion, auf dessen Graphen sich die Punkte befinden. Es werden ganzrationale Funktionen bis zum 5. Grad berechnet. Mit etwas mathematischem Geschick läßt es sich auch auf andere Funktionsstufen umstellen.
Das Programm ist weitgehend Menuegesteuert, zudem ist es abbruchsicher gestaltet mit den Anweisungen: ON WARNING NEXT und ON ERROR 1690.
Es belegt samt seinen Variablen nahezu den gesamten Speicherplatz des TI. Weiterhin benötigen Sie für dieses Programm mindestens das Extended-Basic-Modul, ein Drucker ist empfehlenswert, da Sie alle errechneten Werte zusammenfassend ausdrucken können. Das Programm spricht den Matrixdrucker GP-100 A von Seikosha an. Profis sollte es keine Schwierigkeiten bereiten, das Programm für andere Drucker umzuschreiben.

Leistungen:

Falls Sie dem Rechner eine bereits bekannte Funktion eingegeben haben, erstellt er ihnen eine komplette Kurvendiskussion. Er berechnet Y-Achsenschnittpunkt, Nullstellen, Extrem-Punkte und Wendepunkte. Der Computer berechnet die Fläche unter der Kurve, innerhalb eines variablen X-Achsenintervalls. Eine graphische Darstellung ist ebenfalls integriert, wobei Sie die Abzissen- und Ordinatenbereiche selbst festlegen.

Mit wenigen Punkten zur Funktion

Sollten nur einige Punkte des Graphen bekannt sein, muß das Programm zuerst die Koeffizienten der gesuchten Funktion berechnen. Anm.: Ohne den Computer würde dieses einen gewaltigen Zeitaufwand von vielen Stunden darstellen. Um die Anzahl der nötigen Punkte herauszufinden, addieren Sie zum Grad der Funktion eins.

Beispielberechnung:

Es handelt sich um eine Polynomfunktion 5. Grades. Sie benötigen hier also genau sechs Punkte.
Die Punkte seien [P (x:y)]:
P1(-51:100) P2(-3:-10) P3(-2:-2) P4(2:10) P5(4:-90) P6(7:-10)
Nun wissen Sie [f(x)=y]:
f(-5)=100 f(-3)=-10 f(-2)=-2 f(2)=10 f(4)=-90 f(7)=-10
Jetzt brauchen Sie eine Funktion, dessen Graph durch die festgelegten Punkte führt. Sie kennen die Normalform für ganzrationale Funktionen 5. Grades.

Sie lautet:
f(x)=a*x 5*x 4c*x 3d*x 2+e*x+f
Das kann auch so geschrieben werden:
x 5*a+x 4*b+x 3*c+x 2*d+x*e+f=y
Sie haben sechs verschiedene X-Werte und die daraus resultierenden Y-Werte. Sie setzen nun sechsmal die X-Werte in jeweils einer neuen Gleichung ein und setzen diese gleich dem dazugehörigen Y-Wert.

Das sieht dann so aus:
1. Gleichung:
-3125a+625b-125c+25d-5e+f=100
2. Gleichung:
-243a+81b-27c+9d-3e+f=-10
3. Gleichung: -32a+16b-8c+4d-2e+f=-2
4. Gleichung: 32a+16b+8c+4d+2e+f=10
5. Gleichung:
1024a+256b+64c+16d+4e+f=-90
6. Gleichung:
16807a+2401b+343c+49d+7e+f=-10

Um nun a, b, c, d, e und f zu ermitteln, müssen alle Gleichungen nach dem Gauschen Eliminationsverfahren aufgelöst werden. Diese zeitaufwendige Aufgabe übernimmt der Computer.
Er fragt nach der Anzahl der Unbekannten. Sie müssen hier eine >6< eintippen. Sie geben danach die Koeffizienten der Gleichungen ein. Es wird eine Gleichung nach der anderen eingegeben. Der Rechner fragt immer zuerst nach den Koeffizienten der jeweiligen Gleichung [Gl. 1/Koef. 3=].
Dann fragt er nach dem f(x)-Wert der Gleichung [Gl. 1/f(x) =].
Geben Sie bei jeder Frage nur einen Wert ein!

Die Koeffizientenmatrix:

	Koef. 1	Koef. 2	Koef. 3	Koef. 4	Koef. 5	Koef. 6		f(x)
Gleichung 1	-3125	625	-125	25	-5	1	:	100
Gleichung 2	-243	81	-27	9	-3	1	:	-10
Gleichung 3	-32	16	-8	4	-2	1	:	-2
Gleichung 4	32	16	8	4	2	1	:	10
Gleichung 5	1024	256	64	16	4	1	:	-90
Gleichung 6	16807	2401	343	49	7	1	:	-10

Nach 40 Minuten Rechenzeit wird folgendes Ergebnis angezeigt:

a = .012962963
b = .2767195767
c = -1.616137566
d = -8.94021164
e = 9.257142857
f = 35.33333333

Somit lautet die Funktionsgleichung:

f(x) = .012962963*x^5+.2767195767*x^4-1.616137566*x^3
-8.94021164*x^2+9.257142857*x+35.33333333

Falls Sie sich das Ergebnis nicht ausdrucken lassen, müssen Sie sich die Werte unbedingt notieren!
Der Rechner führt Sie jetzt mittels verschiedener Menues durch das Programm:

Beschreibung des Hauptmenues

Nach dem Programmstart erscheint ein 5 Punkte umfassendes Menue auf dem Monitor.

1. Unter Punkt eins geben Sie dem Rechner Ihre Funktionsgleichung ein. Wenn Sie Taste >1< drücken, gelangen Sie in ein Untermenue, drücken Sie hier am besten die >5<.
2. Unter Punkt zwei weist der Rechner darauf hin, daß er ausschließlich für ganzrationale Funktionen programmiert ist.
   Falls Sie mit anderen Funktionstypen arbeiten wollen, müssen Sie folgendes ändern:
   Die Zeilen 550→560:
   Hier werden die Stammfunktion und die drei Ableitungen definiert.
   Y(X)=Stammfunktion
   AB1=1. Ableitung
   AB2=2. Ableitung
   AB3=3. Ableitung
   Die Zeilen 610→660:
   Hier wird aus den Funktionskoeffizienten ein String gebildet, um die Funktion jederzeit vor Augen zu haben. Haben Sie einen anderen Funktionstyp definiert, so müssen Sie die Stringausdrücke entsprechend umändern.
3. Punkt drei sollten Sie immer dann drücken, wenn Sie die Funktion schon eingegeben haben und mit ihr arbeiten wollen. Sie gelangen in ein Untermenue. (Beschreibung folgt!)
4. Wenn Sie alle berechneten Werte auf dem Bildschirm sehen wollen, so drücken Sie Taste >4<. Ein weiteres Menue sagt, was Sie zu tun haben.
5. Falls Sie die Funktion noch nicht kennen und nur einige Punkte zur Verfügung haben, können Sie diesen Punkt wählen.
   Das angewählte Unterprogramm berechnet die Koeffizienten Ihrer Funktion. (siehe Beschreibung!)
   Mit diesem Unterprogramm können Sie zudem lineare Gleichungssysteme mit bis zu 10 Unbekannten berechnen.

Beschreibung des Untermenues:

Die Unterprogrammpunkte 0→8 sind eindeutig. Falls notwendig beschreibt sich das Programm von selbst.

6. Haben Sie Punkt >6< gewählt, so erscheint eine Eingabemaske. Ist die hier definierte Grafik erstellt und Sie wollen mit anderen Achsenabschnitten arbeiten, so drücken Sie die Gleichheitstaste. Sie gelangen dann wieder zur Eingabemaske.
7. Unter Punkt sieben gelangen Sie mit der '='-Taste aus der Tabelle.
8. Nachdem die Funktion vollständig diskutiert ist können Sie die Kurvendiskussion ausdrucken lassen.

```
1
100 OPTION BASE 1 :: DIM NL(5),MA(5),MI(5),WN(5),U(10,10),BF(10),CF(10),AL(10)::
 OPEN #1:"RS232.DA=8" :: ON WARNING NEXT :: ON ERROR 1690
110 CALL CLEAR :: CALL SCREEN(16):: FOR I=0 TO 12 :: CALL COLOR(I,16,5):: NEXT I
 :: CALL COLOR(13,10,5,14,2,5):: GOSUB 1390 :: DEF @$(I)=CHR$(I):: DEF S$(I)=STR
$(I)
120 SY=0 :: DISPLAY AT(1,1)ERASE ALL BEEP:" komplette Funktionsanalyse  "&RPT$(@
$(127),4)&@$(129)&"(bis zum 5.Grad)"&@$(128)&RPT$(@$(127),4):: CALL HCHAR(3,9,13
0,16)
130 CALL VCHAR(1,32,127,48):: DISPLAY AT(6,1):"1] Eingabe der Koeffizienten" ::
CALL HCHAR(7,6,127,25)
140 DISPLAY AT(10,1):"2] Funktion anderen Typs" :: CALL HCHAR(11,6,127,21):: DIS
PLAY AT(14,1):"3] "&@$(129)&"Funktion beibehalten"&@$(128)
150 CALL HCHAR(15,6,127,22)
160 DISPLAY AT(18,1):"4] Wiederholung der Werte" :: CALL HCHAR(19,6,127,22):: DI
SPLAY AT(22,1):"5] lineares Gleichungssystem" :: CALL HCHAR(23,6,127,25)
170 CALL KEY(0,K,S):: IF S=0 THEN 170 ELSE IF K=51 THEN 500 ELSE IF K=49 THEN SY
=1 :: GOTO 500 ELSE IF K=52 THEN 1250 ELSE IF K=53 THEN 180 ELSE IF K=50 THEN 48
0 ELSE 170
180 DISPLAY AT(1,1)ERASE ALL BEEP:" lineares Gleichungssystem:  "&RPT$(@$(127),2
6): : : :"Analyse bis zum zehnten Grad"&RPT$(@$(127),28)
190 DISPLAY AT(12,1):" Moechten Sie einen Drucker  benutzen ? (J/N)   "&RPT$(@$(
127),7)
200 CALL KEY(0,K,S):: IF S=0 THEN 200 ELSE IF K=74 THEN DR=1 ELSE IF K=78 THEN D
R=0 ELSE 200
210 DISPLAY AT(18,1):"Anzahl der Unbekannten :" :: ACCEPT AT(18,26)BEEP SIZE(-2)
:W :: IF W<2 OR W>10 THEN 210
220 CALL CLEAR :: PRINT :"Eingabe des Gleichungs-":"systems:":
230 FOR P=1 TO W :: FOR S=1 TO W :: INPUT "Gl."&S$(P)&"/Koef."&S$(S)&"=":U(P,S):
: NEXT S :: INPUT "Gl."&S$(P)&"/f(x)=":AL(P):: PRINT :: NEXT P :: CALL CLEAR
240 A$="Gleichungssystem mit "&S$(W)&" Unbekannten:"
250 PRINT SEG$(A$,1,20):SEG$(A$,22,LEN(A$)-21): : :: IF DR=1 THEN PRINT #1: : :A
$:RPT$("=",LEN(A$)): :
260 FOR P=1 TO W :: IF DR=1 THEN PRINT #1: :"Koeffizienten der "&S$(P)&". Gleich
ung:":TAB(5);
270 FOR S=1 TO W :: PRINT INT(U(P,S));:: IF DR=1 THEN PRINT #1:U(P,S);
280 NEXT S :: PRINT "|";AL(P):: IF DR=1 THEN PRINT #1:"|";AL(P)
290 NEXT P :: PRINT :: IF DR=1 THEN PRINT #1
300 FOR N=W TO 2 STEP -1 :: GOSUB 380 :: IF DE=0 THEN 350
310 T=DE :: FOR P=1 TO N :: CF(P)=U(P,N):: U(P,N)=AL(P):: NEXT P :: GOSUB 380 ::
 V(N)=DE/T :: FOR P=1 TO N-1 :: AL(P)=AL(P)-V(N)*CF(P):: NEXT P :: NEXT N
320 V(1)=AL(1)/U(1,1):: A$="Die Loesung der Gleichung " :: B$="lautet:" :: PRINT
 A$:B$: : :: IF DR=1 THEN PRINT #1: :A$;B$: :
330 FOR I=1 TO W :: PRINT @$(96+I);"=";V(I):: IF DR=1 THEN PRINT #1:TAB(6);@$(96
+I);"=";V(I)
340 V(I)=0 :: NEXT I :: GOTO 360
350 A$="Es gibt keine eindeutige" :: B$=" Loesung der Gleichung." :: PRINT "  ";
A$:"  ";B$: : : : : : :: IF DR=1 THEN PRINT #1:A$;B$
360 PRINT :: DISPLAY AT(24,1):"neue Gleichung ? (J/N) :"
370 ACCEPT AT(24,26)SIZE(-1):A$ :: IF A$="J" THEN 180 ELSE IF A$="N" THEN CALL C
LEAR :: GOTO 120 ELSE 370
380 DE=0 :: CE=1 :: FOR I=1 TO N :: CE=CE*I :: BF(I)=I :: NEXT I
390 FOR R=1 TO CE STEP 2 :: O=N-2 :: K=0 :: FOR I=1 TO N-1 :: FOR Q=I+1 TO N ::
IF BF(I)<BF(Q)THEN 400 ELSE K=K+1
400 NEXT Q :: NEXT I :: GOSUB 470 :: H=BF(N):: BF(N)=BF(N-1):: BF(N-1)=H :: K=K+
1 :: GOSUB 470
410 IF O=0 THEN RETURN ELSE BF(O)=BF(O)+1
420 FOR I=1 TO O-1 :: IF BF(I)=BF(O)THEN 410
430 NEXT I :: IF BF(O)<=N THEN 440 ELSE O=O-1 :: GOTO 410
440 H=O :: FOR I=1 TO N :: FOR Q=1 TO O :: IF BF(Q)=I THEN 460
450 NEXT Q :: H=H+1 :: BF(H)=I
460 NEXT I :: NEXT R :: RETURN
470 L=1 :: FOR I=1 TO N :: L=L*U(I,BF(I)):: NEXT I :: DE=DE+(-1)^K*L :: RETURN
480 GOSUB 1350 :: DISPLAY AT(5,1):"Sie muessten dann Statementsin den Zeilen: 55
0-560 sowiein den Zeilen: 610-660 etwasveraendern."
```

```
490 DISPLAY AT(12,1):"In der Dokumentation finden Sie Informationen darueber." :
: GOTO 1220
500 GOSUB 1350 :: DISPLAY AT(4,2):"Menue:":" "&RPT$(@$(127),6)&RPT$(" ",14)&"0] 
Exit"
510 DISPLAY AT(7,1):"1] Y-Achsenabschnitt": :"2] Nullstellen": :"3] Extremstelle
n": :"4] Wendestellen"
520 DISPLAY AT(15,1):"5] 9esamte Kurvendiskussion": :"6] 9raPhische Darstellun9"
: :"7] Wertetabelle": :"8] Flaechenberechnun9": :"9] ComPuterbericht.....>5< ?"
530 CALL KEY(0,T,S):: IF S=0 OR T<48 OR T>57 THEN 530
540 CALL CLEAR :: IF SY=0 THEN 670
550 DEF Y(X)=A*X^5+B*X^4+C*X^3+D*X^2+E*X+F :: DEF AB1=5*A*X^4+4*B*X^3+3*C*X^2+2*
D*X+E :: DEF AB2=20*A*X^3+12*B*X^2+6*C*X+2*D
560 DEF AB3=60*A*X^2+24*B*X+6*C :: F$="="
570 DISPLAY AT(2,1):"Ein9abe der Koeffizienten  einer 9anzrationalen         Fun
ktion mit der Form:" :: CALL HCHAR(1,3,127,25):: CALL HCHAR(5,3,127,25)
580 DISPLAY AT(7,1):"f(x)=a*x^5+b*x^4+c*x^3+d*x^2+e*x+f":"":"":"a =":"":"b =":""
:"c =":"":"d =":"":"e =":"":"f ="
590 ACCEPT AT(11,5)BEEP:A :: ACCEPT AT(13,5)BEEP:B :: ACCEPT AT(15,5)BEEP:C :: A
CCEPT AT(17,5)BEEP:D :: ACCEPT AT(19,5)BEEP:E :: ACCEPT AT(21,5)BEEP:F
600 CALL CLEAR :: MX,MN,NU,WE=0
610 IF A<>0 THEN F$=F$&S$(A)&"*x^5"
620 IF B>0 THEN F$=F$&"+"&S$(B)&"*x^4" ELSE IF B<>0 THEN F$=F$&S$(B)&"*x^4"
630 IF C>0 THEN F$=F$&"+"&S$(C)&"*x^3" ELSE IF C<>0 THEN F$=F$&S$(C)&"*x^3"
640 IF D>0 THEN F$=F$&"+"&S$(D)&"*x^2" ELSE IF D<>0 THEN F$=F$&S$(D)&"*x^2"
650 IF E>0 THEN F$=F$&"+"&S$(E)&"*x" ELSE IF E<>0 THEN F$=F$&S$(E)&"*x"
660 IF F>0 THEN F$=F$&"+"&S$(F)ELSE IF F<>0 THEN F$=F$&S$(F)
670 PRINT "Funktion: f(x)"&F$: :
680 ON T-47 GOTO 120,690,700,810,960,690,1090,1360,1060,1520
690 PRINT "Y-Achsenabschnitt:":"Py=(0|"&S$(Y(0))&")" :: PRINT :: IF T<>53 THEN 1
220
700 GOSUB 1240 :: GOTO 720
710 S=S/10 :: V1=V2 :: V3=V4
720 FOR X=V1 TO V3 STEP S :: V4=X
730 IF Y(X)=0 OR S=1E-9 THEN 750 ELSE IF SGN(V5)=0 THEN 740 ELSE IF SGN(Y(X))<>S
GN(V5)THEN 710
740 V5=Y(X):: V2=V4 :: NEXT X :: GOTO 770
750 PRINT "Nullstelle bei x=";X :: NU=NU+1 :: NL(NU)=X :: CALL SOUND(50,420,0)
760 X=X+.5 :: V1=X :: V3=20 :: K=K+1 :: V5=0 :: S=1 :: GOTO 720
770 IF K<>0 THEN 790
780 PRINT "keine Nullstellen!"
790 IF T<>53 THEN 1220
800 PRINT
810 GOSUB 1240 :: GOTO 830
820 S=S/10 :: V1=V2 :: V3=V4
830 FOR X=V1 TO V3 STEP S :: V4=X
840 IF AB1=0 OR S=1E-8 THEN 860 ELSE IF SGN(V6)=0 THEN 850 ELSE IF SGN(AB1)<>SGN
(V6)THEN 820
850 V6=AB1 :: V2=V4 :: NEXT X :: GOTO 920
860 IF AB2>=0 THEN 880
870 PRINT "relatives Maximum bei":"("&S$(X)&"|"&S$(Y(X))&")" :: MX=MX+1 :: MA(MX
)=X :: GOTO 900
880 IF AB2=0 THEN 910
890 PRINT "relatives Minimum bei":"("&S$(X)&"|"&S$(Y(X))&")" :: MN=MN+1 :: MI(MN
)=X
900 CALL SOUND(50,420,0)
910 K=K+1 :: X=X+.1 :: V1=X :: V3=20 :: V6=0 :: S=1 :: GOTO 830
920 IF K<>0 THEN 940
930 PRINT "keine Extremstellen!"
940 IF T<>53 THEN 1220
950 PRINT
960 GOSUB 1240 :: GOTO 980
970 S=S/10 :: V1=V2 :: V3=V4
980 FOR X=V1 TO V3 STEP S :: V4=X
990 IF AB2=0 OR S=1E-9 THEN 1010 ELSE IF SGN(V7)=0 THEN 1000 ELSE IF SGN(AB2)<>S
GN(V7)THEN 970
```

```
1000 V7=AB2 :: V2=V4 :: NEXT X :: GOTO 1040
1010 IF AB3=0 THEN 1030
1020 PRINT "Wendestelle bei":"("&S$(X)&"|"&S$(Y(X))&")" :: WE=WE+1 :: WN(WE)=X :
: CALL SOUND(50,420,0)
1030 K=K+1 :: X=X+.1 :: V1=X :: V3=20 :: V7=0 :: S=1 :: GOTO 980
1040 IF WE=0 THEN PRINT "keine Wendestellen!": :
1050 GOTO 1220
1060 PRINT "Flaechenberechnung:": : :: FL,V8,V9=0 :: INPUT "untere Grenze: Xu=":
G1 :: INPUT "obere  Grenze: Xo=":G2 :: QQ=Y(G1)+Y(G2)
1070 FOR Q=G1+.02 TO G2-.02 STEP .02 :: V8=V8+Y(Q):: NEXT Q :: V8=V8*2
1080 FOR R=G1+.01 TO G2-.01 STEP .02 :: V9=V9+Y(R):: NEXT R :: V9=V9*4 :: FL=(QQ
+V9+V8)/300 :: PRINT :"Flaeche unter dem Graphen:  A =";FL:"" :: GOTO 1220
1090 DISPLAY AT(1,1)ERASE ALL:"graphische Darstellung:":"":"":"f(x)"&F$ :: CALL
HCHAR(2,3,127,23)
1100 DISPLAY AT(9,1):"untere Grenze (X-Achse)      "&RPT$(@$(127),23):"Xu =":"":"
obere  Grenze (X-Achse)      "&RPT$(@$(127),23):"Xo ="
1110 DISPLAY AT(17,1):"untere Grenze (Y-Achse)      "&RPT$(@$(127),23):"Yu =":"":
"obere  Grenze (Y-Achse)      "&RPT$(@$(127),23):"Yo ="
1120 ACCEPT AT(11,6)BEEP:A1 :: ACCEPT AT(15,6)BEEP:B1 :: ACCEPT AT(19,6)BEEP:C1
:: ACCEPT AT(23,6)BEEP:D1 :: CALL CLEAR
1130 P1=INT((24/(ABS(D1)+ABS(C1)))*ABS(D1))
1140 P2=(32/(ABS(B1)+ABS(A1)))*ABS(A1)+1 :: P3=0
1150 IF P1<1 OR P1>24 OR P2<1 OR P2>32 THEN 1090
1160 DISPLAY AT(24,1):"$="&S$(INT(((B1-A1)/30)*100+.5)/100)&" #="&S$(INT(((D1-C1
)/23)*100+.5)/100)
1170 CALL HCHAR(P1,1,36,31):: CALL VCHAR(1,P2,35,23):: CALL HCHAR(P1,P2,37):: CA
LL HCHAR(P1,31,39):: CALL HCHAR(1,P2,38)
1180 FOR I=A1 TO B1 STEP (ABS(B1)+ABS(A1))/32
1190 P4=P1-INT((24/(ABS(C1)+ABS(D1)))*Y(I)):: P3=P3+1 :: IF P3>32 THEN 1210 ELSE
 IF P4<=0 OR P4>24 THEN 1200 ELSE CALL HCHAR(P4,P3,143)
1200 NEXT I
1210 CALL KEY(0,K,S):: IF S=0 THEN 1210 ELSE IF K=61 THEN 1090
1220 DISPLAY AT(24,1)SIZE(13):"Menue?  (J/N)"
1230 CALL KEY(0,K,S):: IF S=0 THEN 1230 ELSE IF K=74 THEN CALL CLEAR :: GOTO 120
 ELSE IF K=78 THEN CLOSE #1 :: END ELSE 1230
1240 S=1 :: V5,V6,V7,K=0 :: V1=-26.12 :: V3=26 :: RETURN
1250 DISPLAY AT(1,7)ERASE ALL:"Kurvendiskussion:" :: CALL HCHAR(3,1,61,32):: DIS
PLAY AT(6,1):"1] Nullstellen":RPT$(@$(127),14)
1260 DISPLAY AT(9,1):"2] Extremstellen":RPT$(@$(127),16)
1270 DISPLAY AT(12,1):"3] Wendestellen":RPT$(@$(127),15): :"4] Flaeche unter dem
 Graphen"&RPT$(@$(127),28)
1280 CALL KEY(0,K,S):: IF S=0 OR K<49 OR K>52 THEN 1280
1290 GOSUB 1350 :: DISPLAY AT(5,1):"Funktion: f(x)"&F$ :: ON K-48 GOTO 1300,1310
,1330,1340
1300 FOR I=1 TO NU :: DISPLAY AT(I+9,1):"Nullstelle x=";NL(I):: NEXT I :: GOTO 1
220
1310 DISPLAY AT(10,1):"rel. Maximum:" :: FOR I=1 TO MX :: DISPLAY AT(I+10,1):"("
:MA(I);"|";Y(MA(I));")" :: NEXT I
1320 DISPLAY AT(11+MX,1):"rel. Minimum:" :: FOR I=1 TO MN :: DISPLAY AT(I+11+MX,
1):"(";MI(I);"|";Y(MI(I));")" :: NEXT I :: GOTO 1220
1330 DISPLAY AT(10,1):"Wendestelle:" :: FOR I=1 TO WE :: DISPLAY AT(I+10,1):"(";
WN(I);"|";Y(WN(I));")" :: NEXT I :: GOTO 1220
1340 DISPLAY AT(10,1):"Abzissen-,X-Achsenintervall:":"[";S$(G1)&"|";S$(G2)&"]":
:"Flaeche =";FL :: GOTO 1220
1350 DISPLAY AT(1,7)BEEP ERASE ALL:"Kurvendiskussion:" :: CALL HCHAR(3,3,61,28):
: RETURN
1360 GOSUB 1350 :: DISPLAY AT(5,3):"Wertetabelle:": :" X-Werte:","Y-Werte:" :: C
ALL HCHAR(8,3,130,28):: CALL VCHAR(9,16,131,15):: QQ=0
1370 QQ=QQ+1 :: ACCEPT AT(QQ+9,2)BEEP VALIDATE(NUMERIC,"=")SIZE(10):A$
1380 IF QQ>13 OR A$="=" THEN 1220 ELSE DISPLAY AT(QQ+9,17):Y(VAL(A$)):: GOTO 137
0
1390 CALL CHAR(35,"1010107010101010",36,"000000FF101",37,"101010FF1010101",38,"00
10387CFE10101",39,"10181CFE1C181",143,"00183C7E7E3C18")
1400 CALL CHAR(96,"102844007C007C",59,"0010101010101010",127,"000000FF",128,"1030
70FF70301",129,"080C0EFF0E0C08",130,"0000000000000000FF")
```

```
1410 CALL CHAR(131,"101010101010101")
1420 FOR I=65 TO 90 :: READ A$ :: CALL CHAR(I,A$):: NEXT I :: FOR I=97 TO 122 ::
 READ A$ :: CALL CHAR(I,A$):: NEXT I :: FOR I=48 TO 57 :: READ A$ :: CALL CHAR(I
,A$):: NEXT I :: RETURN
1430 DATA 003844447C4444,007844784444 78,00384440404438,00782424242478,007C407840
407C,007C407840404
1440 DATA 003844405C4438,0044447C444444,007C101010107C,007C0404044438,0048506050
4844,0040404040407C
1450 DATA 00446C54444444,004464544C4444,00384444444438,007844447840 4,00384444544
C38,007844447848 44,00384038044438
1460 DATA 007C101010101,0044444444438,00424242422418,00424242425A24,00422418182
442,00412214080808,007E040810207E
1470 DATA 000038043C443E,004040784444 78,00003840404038,0004043C44443C,0000384478
403C,001C207820202,00003C44443C0438
1480 DATA 004040784444 44,00100030101038,00080018080828 10,00202428302824,00202020
20201C,00006854545454,00007844444444
1490 DATA 000038444444438,000078444478404,00003C44443C0404,00002C3020202,00003840
380478,00207820202018,00004444444438,0000444444281
1500 DATA 000044445454 28,00004428102844,00004444 4C340438,00007C0810207C,00384C54
546438,00081808080 81C,00384404384 07C
1510 DATA 00384418044438,00081828487C08,007C4078044438,00384078444438,007C040810
2020,00384438444438,003844443C0438
1520 PRINT :: DISPLAY AT(1,7):"ComPuterbericht:":"       "&RPT$(@$(127),16):;:"Dr
uckersystem initialisierenund >ENTER< druecken!"
1530 CALL KEY(0,K,S):: IF K<>13 THEN 1530
1540 PRINT #1:@$(16);"15";@$(14);"Kurvendiskussion";@$(15);"(ComPuterbericht):":
@$(16);"15";@$(14);RPT$("=",25);@$(15):""
1550 PRINT #1:"Die Funktion hat die Gleichung:" :: PRINT #1:"f(x)"&F$: :
1560 PRINT #1:"":@$(16);"22";@$(14);"Y-Achsenabschnitt:":@$(16);"22";"----------
---------";@$(15):""
1570 PRINT #1:"Der Y-Achsenabschnitt lautet: f(0)=";Y(0):""
1580 PRINT #1:"":@$(16);"28";@$(14);"Nullstellen:":@$(16);"28";"----------";@$
(15):"" :: IF NU=0 THEN 1610
1590 PRINT #1:"":"Nullstelle bei:"
1600 FOR I=1 TO NU :: PRINT #1:@$(16);"17";"x =";NL(I):: NEXT I :: GOTO 1620
1610 PRINT #1:"Es gibt keine Nullstellen!"
1620 PRINT #1:"":"":@$(16);"26";@$(14);"Extremstellen:":@$(16);"26";"----------
---";@$(15):: IF MX=0 THEN PRINT #1:"":"Es gibt kein rel. Maximum!" :: GOTO 1640
1630 PRINT #1:"":"rel. Maximum bei:" :: FOR I=1 TO MX :: PRINT #1:@$(16);"19";"(
";MA(I);"|";Y(MA(I));")" :: NEXT I
1640 IF MN=0 THEN PRINT #1:"":"Es gibt kein rel. Minimum!" :: GOTO 1660
1650 PRINT #1:"":"rel. Minimum bei:" :: FOR I=1 TO MN :: PRINT #1:@$(16);"19";"(
";MI(I);"|";Y(MI(I));")" :: NEXT I
1660 PRINT #1:"":"":@$(16);"27";@$(14);"Wendestellen:":@$(16);"27";"----------
-";@$(15):"" :: IF WE=0 THEN PRINT #1:"Es gibt keine Wendestelle!" :: GOTO 1680
1670 PRINT #1:"Wendestelle bei:" :: FOR I=1 TO WE :: PRINT #1:@$(16);"18";"(";WN
(I);"|";Y(WN(I));")" :: NEXT I
1680 PRINT #1: : :@$(16);"10";RPT$("-",60): : : :: GOTO 1220
1690 CALL SOUND(400,110,0,220,0,330,0):: ON ERROR 1690 :: RETURN NEXT
```

Jetzt überall im Zeitschriften-handel:

Das Softwaremagazin

„Invaded by the Empire“

Invaderspiel für TI-99/4a Extend Basic Joystick-Steuerung

Nach dem Einladen des Programmes, erscheinen nacheinander sich immer wiederholende Titelbilder.

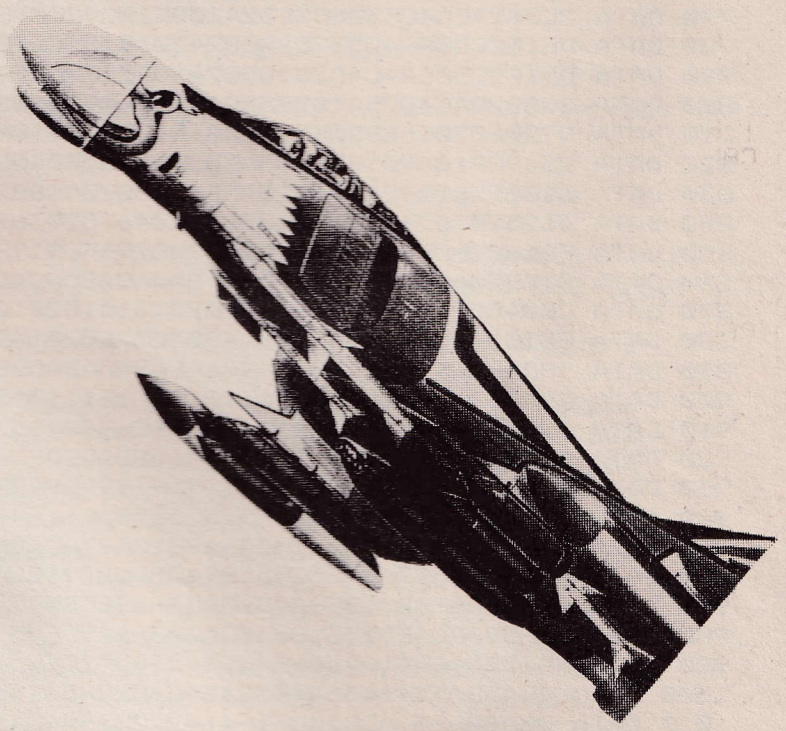
Titelbild 1: Auf dem Bildschirm erscheint der Name des Spieles und die Titelmelodie wird gespielt.

Titelbild 2: Die Punktezählung wird angezeigt.

Titelbild 3: Highscore-Anzeige mit Abfrage der Spieleranzahl.

Wurde hier nach einer bestimmten Zeit keine Taste gedrückt, dann geht es wieder mit Titelbild 1 weiter. Titelbild 1 und 2 können durch Tastendruck jederzeit abgebrochen werden. In diesem Fall erfolgt die Fortsetzung mit Titelbild 3. Nach Festlegung der Spieleranzahl wird das Screen aufgebaut: Sie stehen vor dem Panoramafenster Ihres eigenen Raumschiffes und beobachten die Galaxy. Plötzlich taucht ein feindliches Raumschiff in Ihrem Blickfeld auf, daß immer bedrohlicher und größer werdend auf Sie zu rast (dieser Eindruck wird durch ein dreidimensionales Screen hervorgerufen).

Um nicht mit dem gegnerischen Raumschiff zusammenzustoßen, müssen Sie entweder mit Ihrem eigenen Flugkörper ausweichen oder Sie bringen den Angreifer in den Bereich Ihres Fadenkreuzes, dann haben Sie die Möglichkeit den Gegner zu eliminieren. Wenn Sie es nicht schaffen, innerhalb der Zeitbegrenzung von 30 Sekunden zu bleiben und das Raumschiff zu vernichten, explodiert das eigene Fluggefährt. Wenn es mehr als drei Mal zum Zusammenstoß oder zur Überschreitung der 30-Sekunden-Frist kommt, ist das Spiel beendet.



```
100 ! ************************
110 ! *                      *
120 ! *   COPYRIGHT BY       *
130 ! *                      *
140 ! *  RALF SCHERNEWSKI    *
150 ! *                      *
160 ! *     MARCH 1984       *
170 ! *                      *
180 ! *   INVADED BY THE     *
190 ! *                      *
200 ! *       EMPIRE         *
210 ! *                      *
220 ! ************************
230 HIGH(1)=1000 :: N$(1)="COMPUTER" :: HIGH(2)=600 :: HIGH(3)=300 :: N$(2),N$(3
)=N$(1)
240 !
250 REM DEFINIERUNG VON                ZEICHEN UND FARBEN
260 CALL CLEAR
270 ON WARNING NEXT :: RANDOMIZE
280 ! 48-54
290 DATA 7CFECEDEF6E6FE7C,1838787818181818,7CFE0E3C78E0FEFE,7CFE0E3C3C0EFE7C,1C3
C7CECFEFE0C0C,FEFEC0FCFE06FE7C,7CFEC0FCFEC6FE7C
```

```
300 ! 55-64
310 DATA FEFE060E1C38303,7CFEC67C7CC6FE7C,7CFEC6FE7E06FE7C,00000000006060,0,0,0,
0,0,0
320 ! 65-71
330 DATA 387CEEC6FEFEC6C6,FCFEC6FCFCC6FEFC,7CFEC0C0C0C0FE7C,FCFEC6C6C6C6FEFC,FEF
EC0F8F8C0FEFE,FEFEC0F8F8C0C0C0,7CFEC0CECEC6FE7C
340 ! 72-78
350 DATA C6C6C6FEFEC6C6C6,1818181818181818,FEFE06060606FE7C,C6CEDCF8F8DCCEC6,C0C
0C0C0C0C0FEFE,C6EEFEFED6C6C6C6,C6E6E6F6DECECEC6
360 ! 79-84
370 DATA 7CFEC6C6C6C6FE7C,FCFEC6FEFCC0C0C0,7CFEC6C6D6CEFE7C,FCFEC6FEFCDCCEC6,7CF
EC0FC7E06FE7C,FEFE181818181818
380 ! 85-90
390 DATA C6C6C6C6C6C6FE7C,C6C6EE6C7C38381,C6C6C6C6D6FEFE6C,C6C66C38386CC6C6,C6C6
6C6C38303030,FEFE0C183060FEFE
400 FOR I=48 TO 90 :: READ A$ :: CALL CHAR(I,A$):: NEXT I
410 ! RAUMSCHIFF 6
420 CALL CHAR(140,"00C0C0C0C3C7CCF8F8CCC7C3C0C0C00000030303C3E3331F1F33E3C303030
300")
430 ! EXPLOSION
440 CALL CHAR(116,"80603C1F0F1F7C3B0B0D0E1F1930200040E0E0F8FF7EB8D8BCDE59B8FCFC8
F06")
450 ! RAUMSCHIFF 5
460 CALL CHAR(136,"000000202021223C3C2221202000000000000000040484443C3C44840404000
000")
470 ! RAUMSCHIFF 4
480 CALL CHAR(132,"00000000000001024223140804020100008040201028C4424080000000000
000")
490 ! RAUMSCHIFF 3
500 CALL CHAR(128,"000000000701010202010107000000000000000000C0000080800000C000000
000")
510 ! RAUMSCHIFF 2
520 CALL CHAR(124,"00000000000102050100000000000000000000000000000080A040800000000
000")
530 ! RAUMSCHIFF 1
540 CALL CHAR(120,"0000000000000002030200000000000000000000000000040C040000000000
000")
550 ! GESTELL
560 CALL CHAR(96,RPT$("F",16),97,"FFFEFCF8F0E0C080",98,"0103070F1F3F7FFF")
570 CALL CHAR(99,"FFC3818191A1C3FF")
580 CALL CHAR(104,RPT$("80",8))
590 CALL CHAR(32,"00000000000000000000000103810000000000000000000000000000000000000
00")
600 CALL COLOR(9,15,2,10,2,2)
610 CALL CHAR(100,"80C0E0F0F8FCFEFFFF7F3F1F0F070301")
620 CALL CHAR(102,RPT$("01",8),103,"00000000000000FF")
630 CALL MAGNIFY(3):: FOR I=2 TO 8 :: CALL COLOR(I,5,8):: NEXT I :: CALL COLOR(0
,5,8)
640 CALL CHAR(40,"00000101030307070F0F1F1F3F3F7F7FFFFFFEFEFCFCF8F8F0F0E0E0C0C080
80")
650 CALL CHAR(44,"FFFF7F7F3F3F1F1F0F0F070703030101000008080C0C0E0E0F0F0F8F8FCFCFE
FE")
660 REM *** VORSPANN ***
670 CALL CLEAR :: CALL SCREEN(2):: CALL COLOR(1,16,1):: FOR I=1 TO 24 :: CALL HC
HAR(I,INT(32*RND+1),33):: NEXT I
680 RESTORE 690 :: CALL MAGNIFY(2)
690 DATA I,N,V,A,D,E,D,@,B,Y," ",T,H,E,@,E,M,P,I,R,E
700 J=1 :: K=0 :: FOR I=1 TO 21 :: READ A$ :: IF A$="@" THEN K=16 :: GOTO 760
710 K=K+32 :: J=J+8 :: CALL SPRITE(#I,ASC(A$),5,194,K,-80,0)
720 CALL SOUND(-500,-7,0)
730 CALL POSITION(#I,X,Y):: IF ABS(X-J)>20 THEN 730
740 CALL MOTION(#I,0,0):: CALL LOCATE(#I,J,K)
750 CALL KEY(0,KEY,ST):: IF ST=1 THEN 810
760 NEXT I
770 RESTORE 1860
```

```
780 FOR I=1 TO 161 :: READ A,B,C :: CALL SOUND(-A,B,5,B*2,5,C,5):: CALL KEY(0,KE
Y,ST):: IF ST=1 THEN 800
790 NEXT I
800 CALL SOUND(A,B,0,B*2,0,C,0)
810 CALL MAGNIFY(1)
820 CALL SOUND(-500,-7,0):: CALL MOTION(#1,-50,-50,#2,-50,-35,#3,-50,-20,#4,-50,
0,#5,-50,-20,#6,-50,-35,#7,-50,-50)
830 CALL MOTION(#9,0,-50,#10,0,-50,#11,0,-50,#12,0,50,#13,0,50,#14,0,50)
840 CALL MOTION(#16,50,-50,#17,50,-30,#18,50,-10,#19,50,10,#20,50,30,#21,50,50)
850 IF ST=1 THEN 910
860 FOR I=1 TO 100 :: NEXT I :: CALL DELSPRITE(ALL):: CALL MAGNIFY(3):: DISPLAY
AT(1,1):"PUNKTETABELLE"
870 CALL HCHAR(7,1,32,384)
880 AN=1 :: FOR I=120 TO 140 STEP 4 :: CALL SPRITE(#AN,I,15,(I/4-29)*16+32,30)::
 DISPLAY AT(AN*2+5,10):USING ("## PUNKTE"):150-I :: AN=AN+1 :: NEXT I
890 FOR I=1 TO 300 :: CALL KEY(0,KEY,ST):: IF ST=1 THEN 910
900 NEXT I
910 FOR I=1 TO 100 :: NEXT I
920 CALL DELSPRITE(ALL):: CALL CLEAR :: CALL COLOR(1,5,1):: CALL SCREEN(8)
930 DISPLAY AT(6,1):"   INVADED BY THE EMPIRE"
940 FOR I=1 TO 3 :: DISPLAY AT(15+I*2,8):USING "#: ##### ##########":I,HIGH(I),N
$(I):: NEXT I
950 DISPLAY AT(24,1):"1 ODER 2 SPIELER            _"
960 FOR I=1 TO 200
970 CALL KEY(0,K,S):: IF K=49 OR K=50 THEN 1000
980 NEXT I
990 GOTO 670
1000 CALL HCHAR(24,30,K):: A=K-48
1010 CALL MAGNIFY(3)
1020 REM *** DRUCKEN DES              SPIELBILDSCHIRMS **
1030 CALL CLEAR
1040 PU(1),PU(2)=0
1050 TREFFER=0
1060 RU=1
1070 CALL SCREEN(2)
1080 CALL COLOR(2,15,1)
1090 CALL DELSPRITE(ALL)
1100 FOR I=5 TO 28 :: CALL SPRITE(#I,33,15,(I-4)*5+INT(RND*2),INT(RND*217+16)):
 NEXT I
1110 CALL VCHAR(7,1,96,18):: CALL VCHAR(8,2,96,17):: CALL VCHAR(8,31,96,17):: CA
LL VCHAR(7,32,96,18)
1120 CALL HCHAR(6,1,100):: CALL HCHAR(7,2,100):: CALL HCHAR(7,31,98):: CALL HCHA
R(6,32,98)
1130 CALL HCHAR(1,1,96,64)
1140 CALL HCHAR(19,8,96,18):: CALL HCHAR(21,11,96,13):: CALL HCHAR(8,3,96,28)
1150 CALL VCHAR(21,11,96,4):: CALL VCHAR(21,23,96,4)
1160 RA=3
1170 FOR I=9 TO 17 STEP 2 :: CALL HCHAR(I,RA,44):: CALL HCHAR(I,RA+1,46):: CALL
HCHAR(I+1,RA,45):: CALL HCHAR(I+1,RA+1,47):: RA=RA+1 :: NEXT I
1180 RA=30
1190 FOR I=9 TO 17 STEP 2 :: CALL HCHAR(I,RA,42):: CALL HCHAR(I,RA-1,40):: CALL
HCHAR(1+I,RA-1,41):: CALL HCHAR(1+I,RA,43):: RA=RA-1 :: NEXT I
1200 FOR I=20 TO 24 :: CALL HCHAR(I,27-I,98):: NEXT I
1210 CALL VCHAR(14,16,102):: CALL HCHAR(11,16,102):: CALL HCHAR(12,15,103):: CAL
L HCHAR(12,18,103):: CALL VCHAR(13,17,104,6)
1220 FOR I=20 TO 24 :: CALL HCHAR(I,28-I,96,(I-11)*2):: NEXT I
1230 FOR I=1 TO 5 :: CALL HCHAR(19+I,25+I,100):: NEXT I
1240 CALL HCHAR(20,9,99):: CALL HCHAR(22,7,99,3):: FOR I=22 TO 24 :: CALL HCHAR(
I,24,99,3):: NEXT I
1250 DISPLAY AT(20,10)SIZE(11):"TIME@@@@@@@@"
1260 DISPLAY AT(22,10)SIZE(11):"SCORE@@@@@@@"
1270 DISPLAY AT(24,10)SIZE(11):USING "HIGHS@######":HIGH(1)
1280 DISPLAY AT(24,3)SIZE(6):USING "#@SCH:":3-TREFFER
1290 CALL PUNKTE(PU(),RU)
1300 CALL DELSPRITE(#1)
```

```
1310 DISPLAY AT(1,12)SIZE(9):"SPIELER@";STR$(RU)
1320 CALL SOUND(500,110,0)
1330 CALL SOUND(1,33333,30)
1340 TIME=30
1350 OP=116
1360 ! UNTERPROGRAMM:                    STEURN DER RAUMSCHIFFE
1370 GESCH=INT(RND*(2+PU(RU)/100)+2)*SGN(RND-.5):: GE=INT((2+PU(RU)/100)*RND+1)*
SGN(RND-.5)
1380 CALL SPRITE(#1,120,15,60,121,GE,GESCH)
1390 OP=OP+4 :: IF OP=144 THEN 1480
1400 CALL SOUND(-1000,-3,7)
1410 TIME=TIME-1 :: CALL TIME(TIME):: IF TIME=0 THEN 1480
1420 CALL PATTERN(#1,OP)
1430 GOSUB 1770
1440 CALL POSITION(#1,X1,Y1):: IF Y1<30 OR Y1>226 THEN CALL DELSPRITE(#1,#3):: G
OTO 1350
1450 IF X1>130 THEN CALL DELSPRITE(#1,#3):: GOTO 1350
1460 GOSUB 1770
1470 GOTO 1390
1480 ! UNTERPROGRAMM:                ZUSAMMENSTOSS MIT          RAUMSCHIFF
1490 CALL CLEAR
1500 CALL MAGNIFY(4)
1510 CALL SOUND(-500,-7,0,115,7)
1520 FOR I=1 TO 10 :: CALL SCREEN(16):: CALL SCREEN(10):: CALL SCREEN(12):: NEXT
 I
1530 CALL DELSPRITE(ALL)
1540 CALL MAGNIFY(3)
1550 IF RU<A THEN RU=RU+1 :: GOTO 1070
1560 TREFFER=TREFFER+1
1570 IF TREFFER=4 THEN 1590
1580 IF TREFFER<4 THEN CALL DELSPRITE(ALL):: GOTO 1060
1590 ! SPIEL ENDE
1600 CALL SCREEN(8)
1610 DISPLAY AT(8,11):"GAME OVER"
1620 FOR I=20 TO 5 STEP -5 :: FOR J=I TO 1 STEP -1 :: CALL SOUND(-100,110+J*8,2)
:: NEXT J :: NEXT I
1630 FOR I=5 TO 20 STEP 5 :: FOR J=1 TO I STEP 1 :: CALL SOUND(-100,110+J*8,2)::
 NEXT J :: NEXT I
1640 OW=0 :: PO1=MAX(PU(1),PU(2)):: PO2=MIN(PU(1),PU(2)):: PO=PO1
1650 IF PO<HIGH(3)THEN 1690
1660 IF PO<HIGH(2)THEN HIGH(3)=PO :: GOTO 1680
1670 IF PO<HIGH(1)THEN HIGH(3)=HIGH(2):: HIGH(2)=PO :: N$(3)=N$(2)ELSE HIGH(3)=H
IGH(2):: HIGH(2)=HIGH(1):: HIGH(1)=PO :: N$(3)=N$(2):: N$(2)=N$(1)
1680 IF OW=0 THEN OW=1 :: PO=PO2 :: GOTO 1650
1690 FOR I=1 TO 3 :: IF PU(1)=HIGH(I)THEN DISPLAY AT(15,1):"NAME SPIELER 1 " ::
ACCEPT AT(15,16)VALIDATE(UALPHA)SIZE(10):N$(I)
1700 NEXT I
1710 FOR I=1 TO 3 :: IF PU(2)=HIGH(I)THEN DISPLAY AT(15,1):"NAME SPIELER 2 " ::
ACCEPT AT(15,16)VALIDATE(UALPHA)SIZE(10):N$(I)
1720 NEXT I
1730 DISPLAY AT(10,1):USING "SPIELER 1 ###### PUNKTE":PU(1)
1740 IF A=2 THEN DISPLAY AT(12,1):USING "SPIELER 2 ###### PUNKTE":PU(2)
1750 CALL DELSPRITE(ALL)
1760 GOTO 930
1770 REM STEUERUNG
1780 PU(RU)=PU(RU)+.2 :: CALL PUNKTE(PU(),RU)
1790 CALL JOYST(RU,X,Y):: GESCH=GESCH-X :: GE=GE-Y :: CALL MOTION(#1,GE,GESCH)
1800 CALL KEY(RU,K,S):: IF S=0 THEN RETURN
1810 CALL SOUND(-100,-1,0,115,7):: CALL COLOR(10,4,2):: CALL DISTANCE(#1,89,121,
DIST):: IF SQR(DIST)<(OP-116)/2 THEN CALL COLOR(10,1,1):: RETURN
1820 PU(RU)=PU(RU)+150-OP :: CALL COLOR(10,1,1):: TIME=30
1830 CALL SOUND(-300,-6,0,115,7):: CALL PATTERN(#1,116):: CALL COLOR(#1,10):: FO
R I=1 TO 100 :: NEXT I :: CALL DELSPRITE(#1)
1840 GOTO 1350
```

```
1850 ! DATEN FUER MUSIK
1860 DATA 125,466,22000,125,440,22000,250,440,146,125,466,146,125,440,146
1870 DATA 250,440,146,125,466,146,125,440,146,250,440,146,250,698,146,250,22000,
146,125,698,146,125,659,146
1880 DATA 250,587,146,125,587,146,125,523,146,250,466,146,125,466,146,125,440,14
6
1890 DATA 250,391,195,250,391,195,250,22000,195,125,440,195,125,391,195
1900 DATA 250,391,220,125,440,220,125,391,220,250,391,220,125,440,220,125,391,22
0
1910 DATA 250,391,220,250,659,220,250,22000,220,125,659,220,125,587,220,250,554,
220,125,554,220
1920 DATA 125,466,220,250,440,220,125,440,220,125,391,220,250,349,146,250,349,14
6
1930 DATA 250,22000,146,125,698,146,125,659,146
1940 DATA 250,659,146,250,783,146,250,554,146,250,659,146
1950 DATA 250,587,146,250,440,146,250,22000,146,125,698,146,125,659,146
1960 DATA 250,659,146,250,783,146,250,554,146,250,659,146
1970 DATA 250,587,146,250,698,146,125,659,146,125,587,146,125,523,146,125,466,14
6
1980 DATA 250,22000,146,250,277,146,250,293,146,250,329,146
1990 DATA 250,349,233,125,391,233,125,349,233,250,329,233,250,293,233
2000 DATA 500,277,220,500,293,220
2010 DATA 250,277,220,250,22000,220,500,293,220
2020 DATA 250,277,220,250,22000,220,500,293,220
2030 DATA 250,277,220,250,293,220,250,277,220,250,293,220
2040 DATA 500,277,220,250,22000,220,125,466,220,125,440,220
2050 DATA 250,440,146,125,466,146,125,440,146,250,440,146,125,466,146,125,440,14
6
2060 DATA 250,440,146,250,698,146,250,22000,146,125,698,146,125,659,146
2070 DATA 250,587,146,125,587,146,125,523,146,250,466,146,125,466,146,125,440,14
6
2080 DATA 250,391,195,250,391,195,250,22000,195,125,523,195,125,466,195
2090 DATA 250,466,130,125,523,130,125,466,130,250,466,130,125,523,130,125,466,13
0
2100 DATA 250,466,130,250,783,130,250,22000,130,125,783,130,125,698,130
2110 DATA 250,659,130,125,659,130,125,587,130,250,523,130,125,523,130,125,466,13
0
2120 DATA 500,440,174,250,22000,174,125,698,174,125,659,174
2130 DATA 250,659,146,250,783,146,250,554,146,250,659,146
2140 DATA 250,587,146,250,440,146,250,22000,146,125,698,146,125,659,146
2150 DATA 250,659,146,250,783,146,250,554,146,250,659,146
2160 DATA 250,587,146,250,698,146,125,659,146,125,587,146,125,523,146,125,466,14
6
2170 DATA 250,440,146,250,440,146,250,391,220,250,391,220
2180 DATA 250,329,146,250,22000,146,250,329,146,250,22000,146
2190 DATA 500,293,146
2200 DATA 0,0,0
2210 ! UNTERPROGRAMM:                    ANZAHL DER PUNKTE
2220 SUB PUNKTE(PU(),A)
2230 DISPLAY AT(22,16)SIZE(5):USING "#####":PU(A)
2240 SUBEND
2250 SUB TIME(E)
2260 DISPLAY AT(20,10)SIZE(11):USING ("TIME@@@@@##"):E
2270 SUBEND
```

CALL- und Variablendumps auf dem TI-99/4A

Beabsichtigt man – eigene oder fremde – Basic-Programme zu analysieren, ist es sehr nützlich, wenn man die verwendeten Variablen und CALL-Unterprogramme kennt. Das folgende Maschinenprogramm für den TI-99 mit Extended Basic und 32K-Speichererweiterung stellt Ihnen dafür zwei neue Befehle zur Verfügung, mit denen Variablen- und Unterprogramm-Dumps in Sekundenschnelle auf dem Bildschirm ausgegeben werden können.

Das Funktionsprinzip des Programms ist schnell erklärt: Gibt man RUN ein, so dauert es einige Sekunden, bis ein Basic-Programm wirklich startet. Während dieser Wartezeit legt der Basic-Interpreter im VDP-RAM (Bildschirmspeicher) Tabellen an, die alle Variablen- und Unterprogrammnamen enthalten. Dazu gibt es im CPU-RAM – dies ist eine Art Zero-Page – zwei wichtige Adressen:

– Adresse ›833A (dez. -31942) zeigt auf die Liste mit den CALL-Unterprogrammen.

– Adresse ›833E (dez. -31938) zeigt auf den Beginn der Variablenliste.

Diese Listen sind einander sehr ähnlich und folgendermaßen aufgebaut:

1. Byte: Enthält den sogenannten „Tag-Character", der den Variablentyp bestimmt. ›00 steht für numerische Variable, ›40 für CALL-Unterprogramme und ›80 für Strings. Handelt es sich um ein Datenfeld, wird der Wert um die Anzahl der Dimensionen ergänzt: ›81 ist z. B. ein eindimensionales Stringarray.

2. Byte: Länge des Variablennamens.

3. und 4. Byte: Zeigt auf den nächsten Eintrag in der Liste. Beim letzten Eintrag steht hier ›0000.

5. und 6. Byte: Hier steht ein Pointer auf die Adresse des Variablennamens.

7. und 8. Byte: An dieser Stelle unterscheiden sich die beiden Listen. Bei Variablen steht hier die Speicheradresse ihres Wertes bzw. bei numerischen Feldern die Adresse des ersten Elements, während bei CAL-Unterprogrammen deren Adresse im GROM (Graphics Read Only Memory) gespeichert ist.

Ab dem 9. Byte steht dann der Variablenname; direkt darauf folgt das nächste Listenelement.

Stringarrays weichen von diesem Format etwas ab: Da diese im Gegensatz zu numerischen Feldern normalerweise nicht in der richtigen Reihenfolge im Speicher stehen, sind bei ihnen vor dem Namen für jedes Feldelement noch zwei Bytes für die Adresse des Strings im VDP-RAM eingefügt.

Das Dump-Programm macht sich diese Listen zunutze. Es ist vollständig in Maschinensprache geschrieben und liegt im Low Memory ab Adresse 15100. Somit nimmt es Ihnen keinerlei Basic-Speicher weg und wird auch durch NEW nicht gelöscht. Es ist damit außerdem nach dem Laden jederzeit verfügbar.

Bei der Programmeingabe gibt es drei Wege: Sofern Sie den Editor/Assembler besitzen, können Sie den Quellcode assemblieren. Sie können auch den Objektcode in eine Disketten-Datei des Formats „DISPLAY/FIXED 80" schreiben und das Programm dann jederzeit mit CALL LOAD(„DSK...") einlesen. Der dritte Weg ist der Basic-Loader mit den DATA-Zeilen. In diesem Programm werden auch gleich die neuen Befehle demonstriert.

Steht das Maschinenprogramm nun im Speicher, müssen Sie das zu untersuchende Basic-Programm laden und sofort nach dem Programmstart mit CLEAR (FCTN 4) wieder unterbrechen. Sie können dann folgende Befehle anwenden:

– CALL LINK(„CDUMP") erzeugt eine Liste von sämtlichen benutzten GROM-Unterprogrammen; d. h. selbstdefinierte SUB-Programme werden nicht ausgegeben. Dies bedeutet aber keinen Nachteil, da diese sowieso am Ende des Hauptprogramms stehen müssen und damit leicht zu finden sind.

– CALL LINK(„VDUMP") zeigt alle im Programm oder auch im Direktmodus verwendeten Variablen. Datenfelder werden durch Klammern hinter dem Variablennamen angezeigt, z. B. A$(). Das Programm erwartet übrigens einen sauberen Programmierstil von Ihnen: Aus Gründen der Listenverwaltung müssen sämtliche DEF-Definitionen vor der ersten Variablenzuweisung oder Dimensionierung stehen, da sonst die übrigen Variablen unterschlagen werden. Während der Ausgabe können Sie die Auflistung jederzeit durch Drücken einer beliebigen Taste anhalten. Ein erneuter Tastendruck führt zur Fortsetzung des Listings.

```
100 !   **********************
110 ! **************************
120 ! *                        *
130 ! *  CALL- UND VARDUMP     *
140 ! *                        *
150 ! **************************
160 !   **********************
170 !
180 ! ------------------------------------
190 ! (C) MARTIN KOTULLA 84
200 ! ------------------------------------
210 !
220 CALL CHAR(42,"AA55AA55AA55AA55")
230 CALL CHAR(45,RPT$("0",8)&"FF"&RPT$("0",10)&"10387CFEFE")
240 CALL CLEAR :: CALL HCHAR(2,5,42,25)
250 CALL VCHAR(3,5,42,5):: CALL VCHAR(3,29,42,5)
```

```
260 DISPLAY AT(5,7)SIZE(17):"CALL- UND VARDUMP"
270 CALL HCHAR(8,5,42,25)
280 DISPLAY AT(12,4):"(C) MARTIN KOTULLA 1984"
290 CALL HCHAR(13,6,45,23):: DISPLAY AT(17,4):"PRUEFSUMME BERECHNEN? J"
300 ACCEPT AT(17,26)BEEP SIZE(-1)VALIDATE("JN"):JN$
310 IF JN$="N" THEN 360
320 RESTORE :: FOR I=1 TO 288 :: READ PK :: SUM=SUM+PK :: NEXT I
330 DISPLAY AT(20,5):"DIE DATEN-ZEILEN SIND"
340 IF SUM()19839 THEN DISPLAY AT(22,10):"FEHLERHAFT!" :: END
350 DISPLAY AT(22,11):"KORREKT!" :: RESTORE
360 CALL PEEK(8198,IDCODE):: IF IDCODE()170 THEN CALL INIT
370 FOR I=15100 TO 15382 STEP 2 :: READ A,B :: CALL LOAD(I,A,B):: NEXT I
380 CALL PEEK(8196,A,B):: C=A*256+B-16
390 CALL LOAD(8196,INT(C/256),C-INT(C/256)*256)
400 CALL LOAD(C,86,68,85,77,80,32,59,68,67,68,85,77,80,32,59,54)
410 DISPLAY AT(24,1)ERASE ALL:")CALL(" :: CALL LINK("CDUMP")
420 DISPLAY AT(24,1):")VAR(" :: CALL LINK("VDUMP")
430 CALL SOUND(100,440,0)
440 CALL KEY(0,K,S):: IF S=0 THEN 440 ELSE CALL CLEAR
450 DATA 032,000,040,041,000,004,008,000,181,032,058,068,152,032,172,237
460 DATA 034,094,019,002,181,032,058,069,005,132,002,007,034,090,007,096
470 DATA 040,094,019,003,006,160,059,086,016,004,221,055,221,055,221,055
480 DATA 221,055,152,032,172,237,034,094,022,015,002,224,059,004,192,032
490 DATA 131,058,002,008,059,244,016,007,002,224,059,004,192,032,131,062
500 DATA 019,076,002,008,059,252,004,032,032,024,000,038,004,032,032,040
510 DATA 194,065,010,073,009,201,019,002,007,009,016,001,004,201,009,065
520 DATA 006,152,193,192,005,128,004,032,032,040,192,129,009,130,002,032
530 DATA 000,004,006,160,060,010,192,001,002,001,059,036,004,032,032,044
540 DATA 194,073,019,007,216,160,058,254,059,036,216,160,058,255,059,037
550 DATA 005,194,002,000,002,236,002,003,059,036,208,115,002,033,096,000
560 DATA 004,032,032,032,005,128,006,002,022,248,004,032,032,024,000,038
570 DATA 004,032,032,028,194,096,131,124,034,096,058,252,022,007,004,032
580 DATA 032,028,194,096,131,124,034,096,058,252,022,249,192,007,002,032
590 DATA 000,003,006,160,060,010,192,001,022,185,004,192,216,000,131,124
600 DATA 002,224,131,224,004,096,000,112,152,001,059,001,022,246,004,091
610 DATA 152,001,059,002,019,003,152,001,059,000,022,239,004,091,004,032
620 DATA 032,040,009,129,006,000,004,032,032,040,004,091,007,096,040,100
630 END
```

```
*     ********************************
*   ************************************
*   *                                  *
*   *  CALL- UND VARIABLEN-DUMP        *
*   *                                  *
*   ************************************
*     ********************************
*
*   ------------------------------------
*   (c) Martin Kotulla
*   ------------------------------------
*
*   - CALL LINK("CDUMP")   =  Auflistung der CALL-Aufrufe
*   - CALL LINK("VDUMP")   =  Auflistung der Variablen
*
```

```
        DEF   CDUMP,VDUMP

XMLLNK  EQU   >2018       Equates: Aufruf der ROM-Routinen
KSCAN   EQU   >201C       - Tastaturabfrage
VSBW    EQU   >2020       - Ein Byte in den VDP-RAM schreiben
VSBR    EQU   >2028       - Ein Byte aus dem VDP-RAM lesen
VMBR    EQU   >202C       - Mehrere Bytes aus dem VDP-RAM lesen
SUBPNT  EQU   >833A       - Zeiger auf den Beginn der CALL-Liste
VARPNT  EQU   >833E       - Zeiger auf den Beginn der Variablenliste
ASCII   EQU   >8375       - ASCII-Code der gedrueckten Taste
STATUS  EQU   >837C       - GPL-Status-Byte
GPLWS   EQU   >83E0       - GPL-Workspace
NEXT    EQU   >0070       - Sprung in den GPL-Interpreter
KPRESS  DATA  >2000       Status-Byte, wenn Taste gedrueckt
BRACK1  BYTE  '('
BRACK2  BYTE  ')'
NUMTAG  BYTE  >00         TAG-Character fuer numerische Variable
SUBTAG  BYTE  >04         TAG-Character fuer CALL-Unterprogramme
STRTAG  BYTE  >08         TAG-Character fuer Stringvariable

        EVEN

USRWS   BSS   32          Eigene Workspace-Register
BUFFER  BSS   18          Buffer fuer die Variablennamen

CDUMP   LWPI  USRWS       Eigenen Workspace laden
        MOV   @SUBPNT,R0  Zeiger auf Unterprogrammliste holen
        LI    R8,CHECK1   Geprueft wird das CALL-Tag
        JMP   MAIN        Ins Hauptprogramm springen

VDUMP   LWPI  USRWS       Eigenen Workspace laden
        MOV   @VARPNT,R0  Zeiger auf Variablenliste holen
        JEQ   RETURN      Keine Variablen gespeichert: Ruecksprung
        LI    R8,CHECK2   Geprueft werden die Variablen-Tags

MAIN    BLWP  @XMLLNK     Bildschirm scrollen
        DATA  >26

READLP  BLWP  @VSBR       Den TAG-Character eines Eintrags lesen
        MOV   R1,R9       Fuer Test nach Datenfeld den Tag kopieren
        SLA   R9,4        Die ersten 4 Bits ausblenden
        SRL   R9,12       Die rechten 8 Bits ausblenden
        JEQ   NOARRY      Wenn 0, dann kein Datenfeld

ARRAY   SETO  R9          Flag setzen fuer Datenfeld
        JMP   CONT

NOARRY  CLR   R9          Flag loeschen, da kein Datenfeld

CONT    SRL   R1,4        Um 4 Bits nach rechts schieben
        BL    *R8         Pruefen, ob dieser Character zulaessig ist
        MOV   R0,R7       Anfang des Eintrags fuer naechste Variable
                          sichern

        INC   R0          Zeiger auf das Namenslaengen-Byte
        BLWP  @VSBR       Die Namenslaenge lesen
        MOV   R1,R2       ... und fuer VMBR in Register R2 ablegen
        SRL   R2,8        Den Wert ins LSByte schieben
```

```
         AI    R0,4              Zeiger auf den Pointer auf den Namen
         BL    @VSWR             Dortiges 16 Bit-Wort lesen
         MOV   R1,R0             Wert -> Adresse

VCOPY    LI    R1,BUFFER         VMBR vorbereiten: dort wird
                                 zwischengespeichert
         BLWP  @VMBR             Den Variablen/CALL-Namen dorthin schreiben

CHKARR   MOV   R9,R9             Ist das Flag fuer ein Datenfeld gesetzt?
         JEQ   DSPLAY            Nein - normal weitermachen

DSPARR   MOVB  @BRACK1,@BUFFER(R2) Doch - hinter den Varnamen Klammern
                                 einfuegen

         MOVB  @BRACK2,@BUFFER+1(R2)
         INCT  R2                          und dann zwei Zeichen mehr anzeigen

DSPLAY   LI    R0,748            Zeiger auf unterste Bildschirmzeile
         LI    R3,BUFFER         Zeiger auf den Buffer, in dem die Variable
                                 steht

DISPLP   MOVB  *R3+,R1           Ein Byte aus dem Buffer lesen
         AI    R1,>6000          Bildschirmoffset beruecksichtigen
         BLWP  @VSBW             Und auf den Bildschirm schreiben
         INC   R0                Naechste Bildschirmposition
         DEC   R2                Laengenzaehler dekrementieren
         JNE   DISPLP            Fertig? - Nein: weitermachen

SCROLL   BLWP  @XMLLNK           Den Bildschirm scrollen
         DATA  >26

CHKEY    BLWP  @KSCAN            Tastatur abfragen
         MOV   @STATUS,R9        Wurde eine Taste gedrueckt?
         COC   @KPRESS,R9
         JNE   REDO              Nein - normal weitermachen

EXP2ND   BLWP  @KSCAN            Doch - auf weiteren Tastendruck warten
         MOV   @STATUS,R9
         COC   @KPRESS,R9
         JNE   EXP2ND

REDO     MOV   R7,R0             Die alte VDP-Adresse zurueckholen
         AI    R0,3              Zeiger auf den Pointer fuer den naechsten
                                 Namen
         BL    @VSWR             ... und dieses Wort lesen
         MOV   R1,R0             Wert -> Adresse
         JNE   READLP            Wenn = 0: Ende der Liste, sonst weitermachen

RETURN   CLR   R0                Ruecksprung nach Basic
         MOVB  R0,@STATUS
         LWPI  GPLWS
         B     @NEXT

**************************

CHECK1   CB    R1,@SUBTAG        Steht im VDP-RAM der CALL-Tag?
         JNE   RETURN            Nein - Suche beenden, Ruecksprung nach Basic
         RT                      Doch - Ruecksprung in die Schleife
```

```
CHECK2 CB   R1,@STRTAG          Ist es eine Stringvariable?
       JEQ  CHKRET              Ja - Ruecksprung in die Anzeigeschleife
       CB   R1,@NUMTAG          Ist es eine numerische Variable?
       JNE  RETURN              Nein - Suche beenden, Ruecksprung nach Basic

CHKRET RT                       Ruecksprung in die Anzeigeschleife

********************************

VSWR   BLWP @VSBR               VDP Single Word Write
       SRL  R1,8
       DEC  R0
       BLWP @VSBR
       RT                       Ruecksprung

       END

********************************
OBJEKTCODE:

0011C          A0000B2000B2829B0004B0800A0008A0028A003AB02E0C00087F34BF  0001
A003EBC020B833AB0208C00F8B1007B02E0C0008BC020B833EB134CB02087F2B8F  0002
A0054C0100B0420B2018B0026B0420B2028BC241B0A49B09C9B1302B07097F309F  0003
A006AB1001B04C9B0941B0698BC1C0B0580B0420B2028BC081BC982B02207F2DBF  0004
A0080B0004B06A0C010EBC001B0201C0028B0420B202CBC249B1307BD8A07F2DFF  0005
A0096C0002C0028BD8A0C0003C0029B05C2B0200B02ECB0203C0028BD0737F2E4F  0006
A00ACB0221B6000B0420B2020B0580B0602B16F8B0420B2018B0026B04207F320F  0007
A00C2B201CBC260B837CB2260C0000B1607B0420B201CBC260B837CB22607F2C8F  0008
A00D8C0000B16F9BC007B0220B0003B06A0C010EBC001B16B9B04C0BD8007F2B7F  0009
A00EEB837CB02E0B83E0B0460B0070B9801C0005B16F6B045BB9801C00067F2ACF  0010
A0104B1303B9801C0004B16EFB045BB0420B2028B0981B0600B0420B20287F306F  0011
A011AB045B7FD98F                                                    0012
5003ACDUMP 5004BVDUMP 7FA7AF                                        0013
:       99/4 AS                                                     0014
```

„MODERNE INDUSTRIE-SCHALTUNGEN“

150 professionelle Elektronik-Schaltungen für den Praktiker
Von Horst Pelka, erschienen im FRANCIS-Verlag, München, Preis: 58,– DM

Obwohl heute das Schlagwort „Mikrocomputer“ dominiert – viele Aufgaben können ohne ihn gar nicht mehr gelöst werden –, sind die Arbeitspferde in der Elektronik immer noch Schaltungen, die aus speziellen oder Standard-IC's und ein paar Einzelbauelementen zusammengesetzt sind. Besonders für einfache, gezielte Anwendungen gibt es integrierte Bausteine mit einer erstaunlichen Leistungsfähigkeit.

In dieser Ausgabe sind etwa 150 Schaltungen ausgesucht und zusammengestellt. Die Gliederung wurde nach 8 Hauptanwendungsgebieten vorgenommen. Die Schaltungen sind Applikationsvorschläge großer Halbleiterhersteller, in den meisten Fällen erprobt und leicht nachzubauen.

Der Autor legt einen strengen Maßstab an, bevor ein Applikationsmodell der Halbleiter-Industrie Eingang in die Sammlung fand. Attraktivität, Qualität und Realisierbarkeit, das waren die entscheidenden Kriterien.

Das Spektrum ist weit gespannt. Schwerpunkte wurden auf die Gebiete Messen, Steuern, Regeln und Energieelektronik gelegt. Die Anwendung von integrierten Bausteinen stand im Vordergrund.

Texas Instruments stellt auf der Hannover-Messe komplette Computerfamilie vor

Nur wenige Wochen nach Ankündigung des tragbaren Professional Computers „Pro-Lite" in USA und Europa stellt Texas Instruments mit Business-Pro das „Flaggschiff" der bekannten TIPC-Familie vor.
Als derzeit einziger Hersteller von Mikrocomputern kann Texas Instruments eine „vollständige" Produktpalette offerieren: PCs für mobilen und stationären Einsatz, netzwerkfähig in lokalen und öffentlichen Netzen für Datenübertragung, einige hundert verschiedene Anwendungen und Komplettierung der Systemfamilie durch eigenes Druckerprogramm.
In ca. zwei Jahren ist eine eigenständige neue Generation wirtschaftlicher „Computerleistung" bei Texas Instruments entwickelt worden, deren Anwendungsvielfalt mehr Effizienz und Entscheidungsparameter an den Arbeitsplatz bringen und die Wettbewerbssituation für kleine und mittlere Unternehmen verbessern hilft.

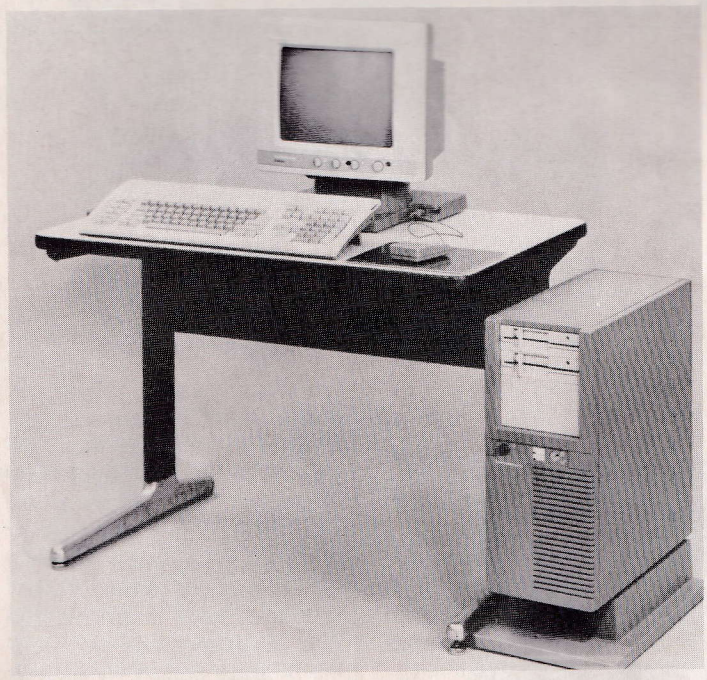
Business-Pro ergänzt die TI Professional Computer-Familie „nach oben" und bietet bei erweiterter Kapazität für Arbeitsspeicher und Massenspeicher gleichzeitig eine höhere Verarbeitungsgeschwindigkeit durch Verwendung eines Hochleistungs-Prozessors 80286.

Das Leistungsspektrum von Business-Pro entspricht den steigenden Anforderungen der Anwender nach Einbindung in vorhandene oder zu implementierende Computer-Netzwerke und nach Systemeigenschaften, die Mehrbenutzer-Betrieb ermöglichen.

Mit 14 Steckplätzen für Systemerweiterungen – Ausbau des Arbeitsspeichers 3 – 15 Mio. Byte, Schnittstellen für Datenübertragung, Steuereinheiten für integrierte oder externe Massenspeicher-Laufwerke (Disketten bis 1,2 MB, Winchester bis 72 MB), Steckmodulen für zusätzlich anschließbare Bildschirm-Arbeitsplätze usw. – läßt sich Business-Pro flexibel schon heute für Applikationen einsetzen, die in Zukunft an Umfang und Komplexität zunehmen werden.

Business-Pro besteht aus drei Systemkomponenten: einem hochauflösenden Bildschirm (monochrom oder Farbe), der ergonomischen Eingabetastatur (getrennte Funktionsblökke) und der Systemeinheit mit allen Steuerfunktionen für das Gesamtsystem. Die Systemeinheit kann „platzsparend" unter einem Schreibtisch untergebracht werden – Business-Pro paßt sich den räumlichen Gegebenheiten eines Arbeitsplatzes an.

Business-Pro ist als TIPC- und IBM PC AT-Version verfügbar und unterstützt somit eine Vielzahl vorhandener Anwendungen. Das Betriebssystem MS-DOS 3.X enthält die Funktionen der vorhergehenden Version 2.X und die notwendigen Erweiterungen zur Unterstützung der Business-Pro-Hardware. Darüber hinaus sind die Möglichkeiten der Dateiverwaltung, Fehlerbeseitigung und -aufzeichnung verbessert, virtuelle Speichermöglichkeiten gegeben und Tastaturanpassungen für nationale Zeichensätze vereinfacht. Das Betriebssystem XENIX 3 gestattet Mehrbenutzer-Betrieb (maximal acht) und wird im 4. Quartal 1985 für Business-Pro angeboten. Auslieferungen der TIPC-/IBM PC AT-Versionen sind für das 3. Quartal vorgesehen.



Der größte Baustein der TI-Computerfamilie

Technische Spezifikationen

Systemeinheit:

- 16-Bit-Prozessor 80286 (6MHz) – 16 Datenbits/24 Adressbits
- Co-Prozessor 80287 (Option)
- RAM-Speicher ab 512 KB (150 ns)
  Ausbau bis 3 MB ohne Belegung eines Erweiterungs-Steckplatzes
  maximaler Ausbau auf 15 MB
- Disketten-Laufwerke 360 KB/1,2 MB
- Winchester-Laufwerke 21/40/72 MB
- ¼-Zoll-Streamer-Tape 60 MB
- 8 plus 6 Erweiterungs-Steckplätze
- Druckeranschluß (Bit-parallel)
- Video-Steuereinheit (Monochrom/Farbe)
- Tastatur-Anschluß
- RS-232-C-(V.24-)Schnittstelle
- LAN-Schnittstelle
- 8/4-Kanal-Schnittstelle (Anschluß weiterer Bildschirm-Terminals)

Fortsetzung S. 74

Load-Programm mit Diskettenkatalog und Generierung von Kleinbuchstaben in Picaschrift

Diskettenversion für den TI-99/4A

Benötigte Konfiguration: TI-99/4A, Ext.-Basic, Diskettenstation

Mit diesem Programm werden alle auf der Diskette befindlichen Programme angezeigt und gestartet.

Alle Texte in den so gestarteten Programmen, werden in Picaschrift, einschließlich ä, ö, ü, ß dargestellt. Voraussetzung ist, daß dieses Programm unter LOAD auf der Diskette abgespeichert wird und alle auf der Diskette befindlichen Programme vom Katalog aus gestartet werden. Entweder automatisch durch die Wahl von Ext.-Basic oder durch RUN „DSKl. LOAD". Danach die entsprechende Programm-Nr. aus der Programmübersicht und ENTER.
Die Umlaute werden durch gleichzeitiges Betätigen der Tasten CTRL&A=ä, CTRL&B=ö, CTRL&C=ü, CTRL&D=ß geschrieben. In den Zeilen bis 120 werden die Picaschrift und Zeichen definiert. Wenn man das Programm startet nachdem bis Zeile 120 eingegeben wurde, kann im weiteren Verlauf sofort mit den neuen Zeichen gearbeitet werden.

In den Zeilen 150–290 werden alle auf der Diskette befindlichen Programme eingetragen, evtl. mit kurzen Erklärungen zum Programm oder es werden Leerzeichen für noch einzutragende Programme freigelassen. In diesen Zeilen entsprechen die Buchstaben ‚E' und ‚O' CTRL&E bzw. CTRL&O.
Ab den Zeilen 370 werden die in der Programmübersicht angezeigten Programme gestartet.

Hier darf nach RUN „DSKl." für x nur der Name eingetragen werden, unter dem das Programm abgespeichert ist.
In allen Programmen, die sich auf dieser Diskette befinden, brauchen also die Buchstaben und Umlaute nicht noch einmal definiert werden. Sie belegen somit in diesen Programmen keinen zusätzlichen Speicherplatz!

```
1o ! Loadprogramm zum Erzeugen v.Kleinbuchstaben
2o ! Markus Spannknebel,Ffm.
3o FOR I=97 TO 114:: READ M$
4o CALL CHAR(I,M$) :: NEXT I
5o FOR I=1 TO 9 :: READ S,M$
6o CALL CHAR(S,M$) :: NEXT I
7o DATA ooooo38o43C443A,oo4o4o4o78444438,ooooo3C4o4o4
   o3C,ooo4o4o43C44443A,ooooo3844784o3C
8o DATA oo18242o7o2o2o2o,oooooo3C443Co438,oo4o4o7844444
   444,ooloool8lololo3C,ooo8ooo8o8o8483o
9o DATA oo4o4o5o6o5o4848,oo3o2o2o2o2o2o78,ooooo7854545
   454,ooooo5864444444,ooooo3C44444438
1oo DATA ooooo3844784o4o,ooooo38443Co4o4,ooooo58242o
    2o2o
11o DATA 116,oo2o2o7o2o2o2o18,117,ooooo444444443A,121,
```

```
    oooooo44443Co438,129,oo44oo38o43C443A,13o,oo44oo3C4
    4444438
12o DATA 131,oo44oo444444443A,132,oo1824282424282o,133,
    ooo8o4o2FFo2o4o8,143,lololololo92543810
1o3 CALL CLEAR
14o DISPLAY AT(9,5):"Programmübersicht" :: FOR I=1 TO 5
    oo :: NEXT I
15o DISPLAY AT(1,2):"1 'E' " :: DISPLAY AT(2,26):"I" !
    'E' entspricht Tasten CTRL&E drücken
16O DISPLAY AT(3,2):"2 'E' " :: DISPLAY AT(4,26):"H"
17o DISPLAY AT(5,2):"3 'E' " :: DISPLAY AT(6,26):"R"
18O DISPLAY AT(7,2):"4 'E' " :: DISPLAY AT(8,26):"E"
19o DISPLAY AT(9,2):"5 'E' " :: DISPLAY AT(12,26):"W"
2oo DISPLAY AT(11,2):"6 'E' " :: DISPLAY AT(14,26):"A"
21o DISPLAY AT(13,2):"7 'E' " :: DISPLAY AT(16,26):"H"
22o DISPLAY AT(15,2):"8 'E' " :: DISPLAY AT(18,26):"L"
23o DISPLAY AT(17,2):"9 'E' " :: DISPLAY AT(2O,26):"?"
24o DISPLAY AT(19,1):"1o 'E'" :: DISPLAY AT(22,26):"'O'
25o DISPLAY AT(21,1):"11 'E'"
3OO DISPLAY AT(23,1):"16 'E' .....E N D E....."
31o ACCEPT AT(24,26)VALIDATE(DIGIT)SIZE(2)BEEP:W
36o ON W GOTO 37o,38o,39o,4oo,41o,42o,43o,44o,45o,46o,4
    7o,52o
37o RUN "DSK1.xxxxxxxxxx" ! Für x gültigen Programmnamen
      einsetzen.
38o RUN "DSK1.xxxxxxxxxx"
39o RUN "DSK1.xxxxxxxxxx"
4oo RUN "DSK1.xxxxxxxxxx"
41o RUN "DSK1.xxxxxxxxxx"
42o RUN "DSK1.xxxxxxxxxx"
43o RUN DSK1.xxxxxxxxxxx"
44o RUN DSK1.xxxxxxxxxxx"
45o RUN DSK1.xxxxxxxxxxx"
46o RUN DSK1.xxxxxxxxxxx"
47o RUN DSK1.xxxxxxxxxxx"
52o END
```

KIRCHNER-Elektronik – Innovative Techniken

In der Automobil-Branche kennt man sie schon seit langem, die „Tuner", die Serienfahrzeuge um- und aufrüsten, um mehr Leistung aus ihnen herauszuholen.
Auch im Computergeschäft haben sich inzwischen einige Firmen darauf spezialisiert, die elektronischen Geräte auf Höchstleistung zu trimmen.
Einer der ersten Computer-Tuner in Deutschland ist der Diplom-Ingenieur Anton Kirchner aus Duisburg-Neudorf. An der Duisburger Universität hat sich Kirchner mit dem „tunen" eines Epson QX-10 einen Namen gemacht. Er schaffte es, die Speicherkapazität des ohnehin schon hochqualifizierten Systems zu verdoppeln, um damit auch den gehobenen Ansprüchen in Forschung und Verwaltung gerecht zu werden.

Gut drei Viertel der von Kirchner-Elektronik verkauften Computer sind getunte Serienprodukte oder Spezialanfertigungen nach Kundenwunsch.

Für den transportablen Einsatz konstruierte Kirchner einen Computer mit kleinstmöglichen Abmessungen. Dieser vollwertige CP/M-Computer wiegt nur knapp 2 kg, mißt 70 x 110 x 170 mm und speichert 800 KB auf seiner Diskette (bei einem Preis von ca. 2000,– DM).

Neben der umfangreichen Hardware-Veränderung erstellt Kirchner selbst eine ganze Reihe kleiner, nützlicher Nebenprogramme, die die Routinearbeiten erleichtern helfen.
Alle Programme haben eines gemeinsam: Sie machen Computer zuverläßiger, anwenderfreundlicher und schneller.
Dennoch sind sie im Preisgefüge für jeden Anwender erschwinglich.

Multicolor-Grafik für den TI-99

Nur wenigen TI-Spezialisten ist bekannt, daß der im TI-99 eingebaute Video-Prozessor TMS9918 (TI-99/4) bzw. TMS9918A (TI-99/4A) neben dem normalen Grafikmodus und der hochauflösenden Grafik auch noch den sog. Multicolor-Modus beherrscht. Das folgende Maschinenprogramm für Extended Basic, 32K-Speichererweiterung und Diskettenlaufwerk macht diese Betriebsart nun auch dem Basic-Programmierer mit vielen einfachen Grafikbefehlen zugänglich.

Im Multicolor-Modus bestehen alle Zeichen nur noch aus einer 4★4-Matrix. Somit ist der Bildschirm in 48 Zeilen zu je 64 Spalten aufgeteilt. In jeder dieser 3072 Bildschirmpositionen kann eine der 16 möglichen Farben angezeigt werden – Sie haben also fast unbegrenzte Farbsetzmöglichkeiten!
Vorher müssen Sie aber noch den Basic-Loader für das Maschinenprogramm abtippen. Um Ihnen dies möglichst einfach zu machen, berechnet das Programm aus jeder DATA-Zeile die Prüfsumme und zeigt eventuelle Tippfehler sofort an. In diesem Fall wird die Zeilennummer ausgegeben und mittels einiger POKEs, die den NUMBER-Modus simulieren, wird die fehlerhafte Zeile angezeigt. Haben Sie das Programm abgetippt und auf Diskette gespeichert, dann sollten Sie sich als Lohn für die Anstrengung die Grafikbeispiele ansehen, die in den Basic-Loader integriert sind.
Wollen Sie nun selber Grafiken programmieren, so stehen Ihnen folgende Befehle zur Verfügung:

– CALL LINK(„HELP") gibt einen Katalog der Grafikbefehle aus, die im folgenden erklärt werden. Dazu liest der Computer die Namenstabelle von 16280 bis 16383 und zeigt die Einträge geordnet auf dem Bildschirm an.

– CALL LINK(„MULTI", **schirmfarbe**) ist der wichtigste Befehl, denn er schaltet den Video-Chip in den Multicolor-Modus um und färbt den Bildschirm entsprechend ein. Natürlich können Sie auch weiterhin dazu CALL SCREEN verwenden. Beim Aufruf dieses Programms wird auch gleich der Bildschirm initialisiert, d. h. mit den entsprechenden Zeichen beschrieben, und die gesamte Farbtabelle wird auf „transparent" gesetzt.

– CALL LINK(„CLEAR") ersetzt den Befehl CALL CLEAR, der nicht mehr funktioniert, und löscht den Bildschirm.

– CALL LINK(„POINT", **zeile,spalte,farbe**) setzt einen Punkt mit der bereits bekannten 4★4-Matrix in der angegebenen Farbe an die gewünschte Bildschirmposition. Hierbei muß der Zeilenwert zwischen 1 und 48, der Spaltenwert zwischen 1 und 64 liegen. Wird ein Farbcode größer als 16 gewählt, so wird der danebenliegende Punkt mit eingefärbt.

– CALL LINK(„HLINE", **zeile,spalte,länge,farbe**) zieht eine horizontale Linie, beginnend an der gegebenen Zeile/Spalte-Position mit der angegebenen Länge in der entsprechenden Farbe.

– CALL LINK(„VLINE", **zeile,spalte,länge,farbe**) macht genau dasselbe – nur eben vertikal.

– CALL LINK(„LDIAG", **zeile,spalte,länge,farbe**) zieht eine Diagonale, die bildlich gesehen von links oben nach rechts unten verläuft.

– CALL LINK(„RDIAG", **zeile,spalte,länge,farbe**) zieht ebenfalls eine Diagonale, nur beginnt diese rechts oben und geht nach links unten.

– CALL LINK(„BLOCK", **zeile,spalte,rechts,runter,farbe**) zeichnet einen Block, dessen linke obere Ecke durch Zeile und Spalte bestimmt ist und der die angegebenen Einheiten nach rechts und nach unten geht.

– CALL LINK(„RECT", **zeile,spalte,rechts,runter,farbe**) zeichnet ein Rechteck. Der Befehl ist genauso aufgebaut wie BLOCK.

– CALL LINK(„INVERT") invertiert alle Farben des Multicolor-Bildschirms bis auf die SCREEN-Farbe; so wird z. B. aus schwarz weiß und umgekehrt.

– CALL LINK(„QUIT", **schirmfarbe**) verläßt den Multicolor-Modus wieder und schaltet in den normalen Grafikmodus zurück. Die Bildschirmfarbe wird entsprechend der Angabe geändert.

– CALL LINK(„RECALL", **schirmfarbe**) ruft den zuletzt bearbeiteten Multicolor-Bildschirm zurück, sofern dieser mit CALL LINK („QUIT") verlassen wurde. Dabei wird auch das SCREEN entsprechend eingefärbt.

```
100 !  ********************
110 ! **********************
120 ! *                    *
130 ! *    MULTICOLOR      *
140 ! *                    *
150 ! **********************
160 !  ********************
170 !
180 ! (C) Martin Kotulla 84
190 ! ----------------------
200 !
210 CALL CHAR(42,"AA55AA55AA55AA55",91,"7E8199A1A199817E")
220 CALL CHAR(45,RPT$("0",8)&"FF")
230 CALL CHAR(128,"DFBF60C0919B959191808080C060BFDF")
240 CALL CHAR(130,"FBFD06030101014951615149030 6FDFB")
250 CALL CLEAR :: CALL HCHAR(2,4,42,26)
260 CALL VCHAR(3,4,42,5):: CALL VCHAR(3,29,42,5)
270 CALL MAGNIFY(3):: CALL SPRITE(#1,128,2,28,45)
280 DISPLAY AT(5,8)SIZE(17):"MULTICOLOR-GRAFIK"
290 CALL HCHAR(8,4,42,26)
300 DISPLAY AT(12,4):CHR$(91)&"  MARTIN KOTULLA 1984"
310 CALL HCHAR(13,6,45,22):: DISPLAY AT(17,7):"DATAs FRUEFEN? J"
320 ACCEPT AT(17,22)BEEP SIZE(-1)VALIDATE("JN"):JN$
330 IF JN$="N" THEN 530
340 FOR I=1 TO 64 :: READ A :: X=X+A :: NEXT I
350 DISPLAY AT(20,1):"PRUEFSUMMEN-DATA:"
360 IF X<>72304 THEN DISPLAY AT(20,19):"FALSCH!" :: END
370 DISPLAY AT(20,19):"KORREKT!"
380 DIM CHECK(64):: RESTORE 670
390 DISPLAY AT(22,1):"PROGRAMM-DATA:"
400 FOR I=1 TO 64 :: READ CHECK(I):: NEXT I
410 FOR I=1 TO 64 :: SUMME=0
420 ZEILE=I*10+750 :: IF ZEILE>1320 THEN ZEILE=ZEILE+30
430 FOR J=1 TO 16 :: READ A :: SUMME=SUMME+A :: NEXT J
440 IF SUMME=CHECK(I)THEN 510
450 CALL SOUND(150,220,0):: DISPLAY AT(22,15):"FEHLER IN";ZEILE
460 Z1=INT(ZEILE/256) | 480 CALL LOAD(-31986,0,10)
470 Z2=ZEILE-Z1*256   | 490 CALL LOAD(-31980,Z1,Z2)
500 CALL LOAD(-31931,1):: END                          | 520 NEXT I
510 DISPLAY AT(22,15)BEEP:ZEILE;"KORREKT!"             | 530 CALL INIT :: RESTORE 760
531 DISPLAY AT(24,1):"ALLES RICHTIG! BITTE WARTEN!"
540 FOR I=9460 TO 10370 STEP 4
550 READ A,B,C,D :: CALL LOAD(I,A,B,C,D):: NEXT I
560 FOR I=16280 TO 16383 STEP 4 :: READ A,B,C,D :: CALL LOAD(I,A,B,C,D)
570 NEXT I :: CALL LOAD(8194,40,130,63,152)
580 CALL HCHAR(17,1,32,256)
590 DISPLAY AT(17,1):"WOLLEN SIE DIE GRAFIK-DEMOS": :"SEHEN? J"
600 ACCEPT AT(19,8)BEEP SIZE(-1)VALIDATE("JN"):JA$
610 IF JA$="J" THEN 1460
620 CALL DELSPRITE(ALL):: CALL CLEAR
630 STOP | 650 ! PRUEFSUMMEN-DATAs:
640 !    | 660 !
670 DATA 0309,0553,1778,1932,1682,0826,1642,1231,1571,0781,1540,1415
680 DATA 0999,1164,0563,1125,1268,0909,1252,1304,1058,1252,1058,1293
690 DATA 1175,1168,1273,1535,1107,1306,0965,1402,1032,0738,1173,0780
700 DATA 0884,0343,1916,0922,1103,0918,1144,0724,1299,1257,0904,1332
710 DATA 1340,1408,1041,0799,1053,1316,1185,1330,0621,1030,1324,1133
720 DATA 1024,0962,1287,0546
730 !
```

```
740 ! PROGRAMM-DATAs:
750 !
760 DATA 000,000,000,002,000,008,000,248,032,000,004,007,000,002,000,006
770 DATA 003,000,000,000,000,042,128,000,064,000,037,068,000,000,037,174
780 DATA 039,134,035,253,038,184,040,183,041,182,061,221,144,000,044,179
790 DATA 045,194,144,000,004,216,061,221,060,191,061,190,062,192,094,197
800 DATA 000,027,000,000,036,254,038,112,212,000,173,181,172,180,169,128
810 DATA 192,000,000,046,000,002,000,053,000,012,000,009,083,203,002,224
820 DATA 036,254,006,160,039,106,002,224,036,254,006,160,039,124,004,192
830 DATA 004,224,037,068,002,009,000,006,192,096,037,068,168,032,036,252
840 DATA 037,068,193,001,006,196,192,192,002,035,000,128,192,129,006,194
850 DATA 002,034,000,032,004,032,032,032,006,193,005,129,128,129,022,001
860 DATA 192,068,006,193,005,128,128,192,017,245,006,009,022,229,004,096
870 DATA 039,092,002,224,036,254,006,160,039,106,004,096,039,092,002,224
880 DATA 036,254,002,001,000,001,006,160,040,036,192,032,131,074,002,032
890 DATA 006,255,004,032,032,048,002,000,224,000,216,000,131,212,002,000
900 DATA 001,224,004,032,032,048,002,000,004,000,004,032,032,048,004,096
910 DATA 0 9,092,002,224,036,254,002,001,000,001,006,160,040,036,200,032
920 DATA 131,074,037,070,006,160,040,036,200,032,131,074,037,074,006,160
930 DATA 040,036,200,032,131,074,037,078,006,032,037,078,004,032,040,052
940 DATA 004,096,039,092,002,224,036,254,006,160,039,172,004,032,040,052
950 DATA 005,160,037,074,006,004,021,250,004,096,039,092,002,224,036,254
960 DATA 006,160,039,172,004,032,040,052,005,160,037,070,006,004,021,250
970 DATA 004,096,039,092,002,224,036,254,006,160,039,172,004,032,040,052
980 DATA 005,160,037,070,005,160,037,074,006,004,021,248,004,096,039,092
990 DATA 002,224,036,254,006,160,039,172,004,032,040,052,005,160,037,070
1000 DATA 006,032,037,074,006,004,021,248,004,096,039,092,002,224,036,254
1010 DATA 006,160,039,222,193,132,004,032,040,052,005,160,037,074,006,006
1020 DATA 021,250,005,160,037,070,200,032,037,076,037,074,006,005,021,242
1030 DATA 004,096,039,092,002,224,036,254,006,160,039,222,193,132,004,032
1040 DATA 040,052,005,160,037,074,006,006,021,250,200,032,037,076,037,074
1050 DATA 193,197,004,032,040,052,005,160,037,070,006,007,021,250,200,032
1060 DATA 037,076,037,074,193,132,004,032,040,052,005,160,037,074,006,006
1070 DATA 021,250,200,032,037,072,037,070,193,197,004,032,040,052,005,160
1080 DATA 037,070,006,007,024,250,016,048,002,224,036,254,002,000,056,000
1090 DATA 004,193,004,032,032,040,005,065,004,032,032,032,005,128,002,128
1100 DATA 062,008,017,247,004,096,039,092,002,224,036,254,004,032,032,024
1110 DATA 000,038,002,007,063,152,002,008,037,062,002,009,000,006,209,183
1120 DATA 002,038,096,000,222,006,006,009,022,250,002,000,002,226,002,001
1130 DATA 037,062,002,002,000,006,004,032,032,036,004,032,032,024,000,038
1140 DATA 005,199,002,135,063,255,017,231,216,032,036,244,131,124,002,224
1150 DATA 131,224,004,096,000,112,002,000,056,000,004,193,004,032,032,032
1160 DATA 005,128,002,128,062,008,017,250,194,139,002,001,000,001,006,160
1170 DATA 040,036,192,032,131,074,002,032,006,255,004,032,032,048,002,000
1180 DATA 232,000,216,000,131,212,002,000,001,232,004,032,032,048,002,000
1190 DATA 004,007,004,032,032,048,004,090,194,139,002,001,000,001,006,160
1200 DATA 040,036,200,032,131,074,037,070,006,160,040,036,200,032,131,074
1210 DATA 037,074,006,160,040,036,193,032,131,074,006,160,040,036,200,032
1220 DATA 131,074,037,078,006,032,037,078,004,090,194,139,002,001,000,001
1230 DATA 006,160,040,036,200,032,131,074,037,070,200,032,131,074,037,072
1240 DATA 006,160,040,036,200,032,131,074,037,074,200,032,131,074,037,076
1250 DATA 006,160,040,036,193,032,131,074,006,160,040,036,193,096,131,074
1260 DATA 006,160,040,036,200,032,131,074,037,078,006,032,037,078,004,090
1270 DATA 004,192,004,032,032,012,004,032,032,024,018,184,005,129,004,091
1280 DATA 037,030,040,056,193,160,037,074,008,022,010,054,193,032,037,070
1290 DATA 161,132,002,038,056,000,008,052,057,032,036,250,161,133,192,006
1300 DATA 193,032,037,078,006,196,004,193,004,032,032,040,195,032,037,074
1310 DATA 004,203,062,224,036,246,195,012,019,003,002,065,240,000,016,003
1320 DATA 002,065,015,000,010,068,160,068,004,032,032,032,003,128,002,000
```

```
1330 !
1340 ! NAMENSTABELLE:
1350 !
1360 DATA 072,069,076,080,032,032,039,028,081,085,073,084,032,032,037,178
1370 DATA 082,069,067,065,076,076,037,090,073,078,086,069,082,084,038,252
1380 DATA 082,069,067,084,032,032,038,168,066,076,079,067,075,032,038,128
1390 DATA 082,068,073,065,071,032,038,100,076,068,073,065,071,032,038,072
1400 DATA 086,076,073,078,069,032,038,048,072,076,073,078,069,032,038,024
1410 DATA 080,079,073,078,084,032,037,230,067,076,069,065,082,032,037,166
1420 DATA 077,085,076,084,073,032,037,082,000,000,000,000,000,000,000,000
1430 !
1440 ! MULTICOLOR-DEMO::
1450 !
1460 ON WARNING NEXT :: CALL DELSPRITE(#1)
1470 FOR P=1 TO 11
1480 ON P GOSUB 1520,1580,1630,1670,1720,1770,1830,2000,2050,2100,2140
1490 CALL SOUND(150,440,0)
1500 CALL KEY(0,X1,X2):: IF X2=0 THEN 1500
1510 NEXT P :: CALL CLEAR :: END
1520 ! **********************
1530 CALL LINK("MULTI",2):: FOR I=0 TO 9.2 STEP .3
1540 CALL LINK("POINT",INT(30*ATN(I)),I*7,14)
1550 CALL LINK("POINT",SQR(ABS(6-I^2))*5,I*7,7)
1560 CALL LINK("POINT",INT((SIN(I)+1)*20),I*7,13)
1570 NEXT I :: RETURN
1580 ! **********************
1590 CALL LINK("MULTI",2):: FOR I=PI+PI/40 TO 3*PI STEP PI/40
1600 A=ABS(ATN(COS(I)*TAN(I))*20)+13 :: B=I*7-15
1610 CALL LINK("POINT",A,B,14):: CALL LINK("POINT",48-A,B,13)
1620 NEXT I :: RETURN
1630 ! **********************
1640 CALL LINK("MULTI",2):: FOR I=1 TO 48
1650 J=J+1 :: IF J>16 THEN J=3
1660 CALL LINK("HLINE",I,2,I,J):: NEXT I :: RETURN
1670 ! **********************
1680 CALL LINK("MULTI",2):: FOR I=1 TO 48
1690 J=J+1 :: IF J>13 THEN J=3
1700 CALL LINK("HLINE",I,2,I,J):: CALL LINK("HLINE",I,64-I,64-I,16-J)
1710 NEXT I :: RETURN
1720 ! **********************
1730 CALL LINK("MULTI",2):: FOR I=1 TO 48
1740 J=J+1 :: IF J>13 THEN J=3
1750 CALL LINK("VLINE",1,I,47,J):: CALL LINK("HLINE",I,1,60,16-J)
1760 NEXT I :: RETURN
1770 ! **********************
1780 CALL LINK("MULTI",2):: FOR L=1 TO 4
1790 FOR I=1 TO 38
1800 J=J+1 :: IF J>16 THEN J=3
1810 CALL LINK("LDIAG",I,38-I,48-I,J):: NEXT I
1820 J=J-2 :: NEXT L :: RETURN
1830 ! ********************
1840 CALL LINK("MULTI",2):: FOR I=28 TO -30 STEP -1
1850 F=F+1 :: IF F>16 THEN F=3
1860 CALL LINK("RDIAG",18,I,28,F):: NEXT I
1870 FOR I=-27 TO 32
1880 F=F+1 :: IF F>16 THEN F=3
1890 CALL LINK("LDIAG",18,I,28,F):: NEXT I
```

```
1900 FOR I=1 TO 20 :: CALL LINK("HLINE",I,I,40,F)
1910 F=F+1 :: IF F>16 THEN F=3
1920 NEXT I
1930 FOR I=1 TO 30 :: CALL LINK("RDIAG",1,I,20,F)
1940 F=F+1 :: IF F>16 THEN F=3
1950 NEXT I
1960 FOR I=1 TO 30 :: CALL LINK("VLINE",I,I,15,F)
1970 CALL LINK("VLINE",32-I,64-I,15,16-F)
1980 F=F+1 :: IF F>13 THEN F=3
1990 NEXT I :: RETURN
2000 ! ************************
2010 CALL LINK("MULTI",2):: FOR I=40 TO 3 STEP -1
2020 F=F+1 :: IF F>16 THEN F=3
2030 CALL LINK("BLOCK",I,I,5,5,F):: CALL LINK("BLOCK",I,56-I,5,5,F)
2040 NEXT I :: RETURN
2050 ! ************************
2060 CALL LINK("MULTI",2):: FOR I=6 TO 40 STEP 2
2070 F=F+1 :: IF F>16 THEN F=3
2080 CALL LINK("RECT",5,10,I,I,F)
2090 NEXT I :: RETURN
2100 ! ************************
2110 CALL LINK("MULTI",2):: FOR I=6 TO 30
2120 J=J+1 :: IF J>16 THEN J=3
2130 CALL LINK("RECT",5,I,I,I,J):: NEXT I :: RETURN
2140 ! ************************
2150 F=3 :: S=6 :: CALL LINK("MULTI",2):: FOR I=1 TO 25
2160 CALL LINK("LDIAG",1,S,8,F):: CALL LINK("RDIAG",9,S+6,7,F)
2170 S=S+2 :: F=F+1 :: IF F>16 THEN F=3
2180 NEXT I
2190 F=3 :: S=S-1 :: FOR I=1 TO 25
2200 CALL LINK("RDIAG",16,S,8,F):: CALL LINK("LDIAG",24,S-6,7,F)
2210 S=S-2 :: F=F+1 :: IF F>16 THEN F=3
2220 NEXT I
2230 S=S+1 :: FOR I=1 TO 25
2240 CALL LINK("LDIAG",31,S,8,F):: CALL LINK("RDIAG",39,S+6,7,F)
2250 S=S+2 :: F=F+1 :: IF F>16 THEN F=3
2260 NEXT I
2270 FOR I=1 TO 10 :: CALL LINK("INVERT"):: NEXT I
2280 RETURN
2290 END ! ************************************************************************
```

Tips und Tricks für den ZX-Spectrum

Spectrum-Programme, die zahlreiche Farbbefehle enthalten, lassen sich beschleunigen, wenn man die Farbwerte durch einen Pokebefehl eingibt. Um dies zu erreichen, gebraucht man die Systemvariable ATTRP. Poked man anschließend eine Zahl zwischen 0–255 ein, so erfolgen die weiteren Ausgaben in der gewählten Farbe. Die Adresse von ATTRP lautet 23693.

Errechnet man vor Beginn der Programmierung die Farbwerte, so können die gesamten INK-, PAPER-, BRIGHT- und FLASHBEFEHLE durch jeweils einen POKE-Befehl ersetzt werden, dies bewirkt nicht nur eine höhere Ablaufgeschwindigkeit, sondern auch eine Speicherplatzersparnis.

Die zu pokenden Werte errechnen sich wie folgt:

Wert für INK + 8 x Wert für PAPER + 64 für BRIGHT 1 + 128 für FLASH 1.

Ein helles, blinkendes Zeichen – gelb auf blauem Grund – besitzt den Wert 206. Also POKE 23693,206.

Der Wert 206 ergibt sich aus: 6 + 8 + 64 + 128 = 206.

Ein Zeichen BRIGHT 0: FLASH 0: INK 4: PAPER 7 besitzt den Wert: 4 + (7x8) + 0 + 0 + 60.

High-Resolution Graphik

Ein Unterprogramm zum Plotten von einzelnen Punkten für den TI-99

Erläuterungen:

Zeile:	Erläuterungen:
40- 90	Programmkopf
100-110	Erklärung
120-170	Dateneingabe
180-210	YMAX, YMIN-Berechnung
220-250	Übergabewerte XO/YO -errechnen
260-270	Tastaturabfrage E = ENDE W = Wiederholt Plot N = Neue Werte
280-320	Berechnung der Punkte im 192 mal 256 Raster
330-340	Hexadezimalumrechnung

Alle „!" können ersatzlos gestrichen werden.
Das Unterprogramm Call Plot kann natürlich auch anderweitig verwendet werden, z. B. zu einem Zeichenprogramm, das auch beiliegt, daß ich aber nicht näher erklären will, da es sich um exakt die Zeilen 280-340 und einer Joystickabfrage handelt. Es kommen also noch 3 Zeilen hinzu. Soll ein Punkt gesetzt werden, braucht man nur den Feuerknopf drücken. Soll der Bildschirm gelöscht werden, „A" drücken.
Das Unterprogramm kann auch noch in viele andere Grafikprogramme eingebunden werden (Zeile 280-340) und braucht dann nur mit CALL PLOT (X,Y-,)„123456789ABCDEF") aufgerufen zu werden. X für die X-Achse (1-256); Y für die Y-Achse (1-192)!!!

Programmbeschreibung:
Nach dem Start des Programms erscheint eine kurze Erklärung. Nach richtigem Start müssen dann der Anfangswert, Endwert, Schrittweite, YMAX, YMIN eingegeben werden. Bei der Sin-Funktion sind das z. B. Anfangswert=O, Endwert=2x-PI=6.28, Schrittweite=.05 (Normal zwischen .1 und .01), YMAX=1, YMIN=-1.
Wenn man die YMAX- und YMIN-Werte nicht im Kopf hat, dann gibt man einfach für beides „O" ein und der Rechner errechnet sich die Werte selbst. Das dauert etwas länger. Das Programm ist aber trotz allem das schnellste Plot-Programm für den TI in Basic.
Das Programm ist gegen falsche Werte weitgehend gesichert, und so kann man mit den YMAX- und YMIN-Werten etwas spielen, und man erhält tolle Verzerrungen. Werden bestimmte Grenzwerte überschritten, wie z. B. der Zeichen-Vorrat oder X- und Y-Werte, so stürzt das Programm nicht ab, sondern macht mit dem nächsten Wert weiter. Der Plot ist dann erstellt, wenn ein akustisches Signal ertönt. Wahlmöglichkeiten sind dann, siehe oben.

Variablenliste:

YMAX	Maximum von Y
YMIN	Minimum von Y
Y1	Y-Effektiv
X,Y	X,Y-Werte
XO,YO	Übergabewerte an SUB PLOT
K1,S1	Keyrückgabevariablen
XE	X-ENDE
XA	X-ANFANG
S	Schrittweite
H§,C§	Stringausdrücke für die CHAR-Definition
Q,K	Variable für Zeichen (ASCII), das gesetzt werden soll
V,W	Annahmevariablen des SUB PLOT
VG,WG,VR,WR,VO,WO	Variablen zur Errechnung der Punkte bzw. der CHARs

```
40 !HIGH RES. PLOT                    2.4 Kbyt
50 !RAINER GAWRIKOW
60 !SOMMERHAUSEN 15
70 !5203 MUCH
80 !TEL.:02245/3983
90 !
100 DISPLAY AT(8,1)ERASE ALL:"HOCHAUFLOESENDER PLOT!":"_____________________":
: :"VON RAINER GAWRIKOW 2/85": : :"!!DIE FUNKTION MUSS IN ZEILE": : :"130 DEFIN
IERT WERDEN. Z.B.:"
110 DISPLAY AT(21,1):"Y=SIN(X),DANN RUN 130!!" :: STOP
120 !EINGABE DER WERTE
130 DEF Y=SIN(X)
140 CALL SCREEN(3):: DISPLAY AT(12,1)ERASE ALL:"ANFANGSWERT ?": :"ENDWERT     ?"
: :"SCHRITTWEITE?": :"YMAX        ?": :"YMIN        ?"
150 ACCEPT AT(12,15)BEEP:XA :: ACCEPT AT(14,15)BEEP:XE :: IF XE<=XA THEN 150
160 ACCEPT AT(16,15)BEEP:S :: IF S<=0 THEN 160
170 ACCEPT AT(18,15)BEEP:YMAX :: ACCEPT AT(20,15)BEEP:YMIN :: IF YMIN>YMAX THEN
170
180 !ERRECHNEN DER GROESSTEN UND KLEINSTEN Y-WERTE
190 ON BREAK NEXT :: ON WARNING NEXT :: IF YMAX=0 AND YMIN=0 THEN 200 ELSE 210
200 FOR X=XA TO XE STEP S :: YMAX=MAX(YMAX,Y):: YMIN=MIN(YMIN,Y):: NEXT X
210 Y1=YMAX-YMIN :: IF Y1=1 THEN Y1=2
220 !ROUTINE ZUM ERRECHNEN DER WERTE IM(192*256) RASTER
230 CALL CLEAR :: FOR X=XA TO XE STEP S
240 X0=((-SGN(XA)*ABS(XA))+X)*(250/(XE-XA)):: Y0=180-ABS(YMIN*(170/Y1))-Y*(170/Y
1):: IF Y0<8 OR Y0>182 THEN 250 ELSE CALL PLOT(Y0,X0,"123456789ABCDEF")
250 NEXT X :: CALL SOUND(-500,500,10)
```

```
260 CALL KEY(0,K,S1):: IF S1=0 THEN 260 ELSE IF K=87 THEN CALL CLEAR :: GOTO 230
 ELSE IF K=78 THEN CALL CLEAR :: CALL CHARSET :: GOTO 140 ELSE IF K=69 THEN CALL
 CLEAR :: CALL CHARSET :: GOTO 100
270 CALL CLEAR :: END
280 !ROUTINE ZUR EINZELPUNKTANSTEUERUNG (192/256)
290 SUB PLOT(V,W,H$)
300 W=INT(W):: WG=INT(W/8):: WR=W-WG*8+1 :: V=INT(V):: VG=INT(V/8):: VR=V-VG*8+1
 :: W0=INT((WR/4.1-INT(WR/4.1))*4.1+.5):: V0=VR*2+INT(WR/4.1)-1
310 CALL GCHAR(VG+1,WG+1,Q):: IF Q=32 THEN K=K+1 :: Q=K+32 :: C$=RPT$("0",16)ELS
E CALL CHARPAT(Q,C$)
320 IF Q>=143 THEN SUBEXIT
330 !HEXADEZIMALUMRECHNUG
340 C$=SEG$(C$,1,V0-1)&SEG$(H$,POS(H$,SEG$(C$,V0,1),1)OR 2^(4-W0),1)&SEG$(C$,V0+
1,15):: CALL CHAR(Q,C$):: CALL HCHAR(VG+1,WG+1,Q):: SUBEND
```

Das Floppybuch zum Apple II

Data-Becker-Verlag, Düsseldorf; Autor: Manfred Czerwinski; ISBN-3-89011-072-X

Die kleinen biegsamen Kunststoffscheiben – Disketten, Floppydisks oder kurz Floppies genannt – sind aus der Welt der Mikrocomputer nicht mehr wegzudenken. Das Floppy-Buch zum Apple II befaßt sich intensiv mit den Einsatzmöglichkeiten dieses Speichermediums. Der Schwerpunkt liegt bei der Behandlung der verschiedenen Dateiformen.

Der Darstellung liegt das Diskettenbetriebssystem DOS 3.3 zugrunde. Im letzten Kapitel wird das neue System ProDOS behandelt; die erweiterten Möglichkeiten und Unterschiede zu DOS werden aufgezeigt. Alle Programme, die Sie in diesem Buch finden, sind unter beiden Betriebssystemen lauffähig.

Sie sollten bereits über solide Grundkenntnisse in BASIC verfügen und mit Applesoft-Befehlen vertraut sein. Die Prinzipien der Programmierung wie beispielsweise das Formulieren von Schleifen und Verzweigungen, die Funktion von Ein- und Ausgabebefehlen oder der Umgang mit Feldern und Zeichenketten werden vorausgesetzt.

Auf dem Gebiet der Dateiverarbeitung benötigen Sie keinerlei Kenntnisse, da wir sozusagen bei „Null" anfangen. Aber auch für Fortgeschrittene, die mit sequentiellen und Random-Dateien schon gearbeitet haben, bieten insbesondere die Kapitel 5 bis 9 neue Möglichkeiten, Alternativen sowie Tips und Tricks.

Sie finden in diesem Buch eine ganze Reihe von Programmen, die Sie direkt in Ihren Apple-Computer eintippen und anwenden können. Wenn Sie beispielsweise eine Kunden-, Artikel- oder Vereinskartei benötigen, können Sie das entsprechende Programm unmittelbar einsetzen. Das Ziel des Buches liegt jedoch **nicht** darin, fertige Programme zum Abtippen zu präsentieren. Vielmehr sollen Sie die **Prinzipien** der Dateiverarbeitung kennen- und verstehen lernen und den **Entwurf der Programme** Schritt für Schritt nachvollziehen können. Diese Vorgehensweise ist zwar für Sie zunächst etwas mühsamer, Sie profitieren aber wesentlich mehr davon: Sie können nämlich die Programme so modifizieren, daß diese für Ihre persönlichen Anwendungen optimal sind. Und Sie sind in der Lage, eigene Programme zu schreiben, welche auf den hier dargestellten Prinzipien beruhen. Aus diesem Grund enthält das Buch zahlreiche Abbildungen zur Verdeutlichung wichtiger Programmiermethoden; ferner sind alle Programme ausführlich dokumentiert. Das Buch ist daher Lehrbuch und Programmsammlung in einem.

Einige Kapitel aus dem Inhaltsverzeichnis:

- Eigenschaften und Anwendungen einer Floppy
- Unsortierte sequentielle Dateien
- Sortierte sequentielle Dateien
- Random-Dateien
- Such- und Sortierverfahren
- Index-Sequentielle Dateien
- Verkette Dateien
- Programmierung eines Literaturverzeichnisses
- Weitere DOS-Kommandos und Besonderheiten
- Das System ProDOS

3½ Zoll Micro-Floppy-Disk zum Mitnehmen!

Für Olivetti M10 und Tandy TRS80-Model 100.

Daß man seinen Computer mit auf die Reise nehmen kann, ist vielen schon eine Selbstverständlichkeit. Aber wenn schon mobiler Computer – dann auch mobiler Speicher. In der Größe sollte natürlich auch alles zusammenpassen.

Damit wären wir auch schon bei der PFD-400. Der netzunabhängigen 3½ Zoll Micro-Floppy für Ihren M10 bzw. Model 100. Klein und handlich mit erstaunlicher Leistung: 3½ Zoll Drive mit Schnittstelle zu Ihrem Computer. Wiederaufladbare Ni-Cad-Batterien sowie Netzanschluß für ein Ladegerät.

Da ist Platz für jede Menge Daten – ca. 360 KB können Sie pro Diskette (formatiert) abspeichern. Die Zugriffszeit: Ganze 6 Millisekunden (Track to Track).

Die PFD-400 schließen Sie ganz einfach an Ihren Computer über das mitgelieferte Kabel an (BUS-Verbindung). Übrigens – Ihre Schnittstellen für Drucker und Modem bleiben frei!

Last but not least ist Ihre Floppy mit einem integrierten Videointerface ausgerüstet. Problemlos schließen Sie hier einen Monitor (80 x 25) oder ein TV-Gerät (40 x 25) an.

2. Elektronik- & Computertage Saar

2. Verkaufs- und Informationsmesse für Elektronik- und Microcomputer vom 6.–8. September 1985 in der Kongreßhalle Saarbrücken

Mit über 60 Ausstellern und fast 7000 Besuchern waren die 1. Elektronik- & Computertage Saar für alle Beteiligten ein großer Erfolg.
Nach der Eröffnung durch den Schirmherrn, dem saarländischen Landtagspräsidenten Albrecht Herold, Vertretern der Ministerien sowie Presse, Rundfunk und Fernsehen, bot sich den Besuchern auf einer Ausstellungsfläche von fast 1100 qm ein repräsentativer Querschnitt durch die gesamte Elektronik- und Microcomputerbranche.

Einen sehr hohen Besucheranteil stellten Vertreter von Handwerk- und Industriebetrieben aus dem Saarland und den benachbarten Regionen dar. Besonders erfreulich war die Tatsache, daß viele Besucher aus Frankreich, Luxemburg und sogar aus Belgien die Elektronik- & Computertage Saar besucht haben.
An diesem positiven Ergebnis hatte sicherlich die Präsens von namhaften Firmen wie APPLE, EPSON, FUJITSU, HP, IBM, KONTRON, OLIVETTI, TOSHIBA, TRIUMPH ADLER, COMMODORE und WANG, um nur einige wenige zu nennen, einen großen Anteil.
Daß diese Firmen auch dieses Jahr ihr Kommen, teilweise mit erheblicher Standvergrößerung, wieder zugesagt haben, ist ein Grundstein für ein noch besseres Resultat.

Neben den kommerziellen Rechnern findet der Interessent eine große Auswahl an Zubehör und Fachliteratur. Aber auch die Elektronik mit modernen Kommunikationstechniken und der Video- und Audiobereich gehören zum Ausstellungsschwerpunkt. Im Foyer West der Kongreßhalle finden unsere jüngeren Besucher ihr „Heimcomputerparadies", mit allem Zubehör, was das Herz begehrt.
Die Elektronik- & Computertage Saar werden auch 1985 nach dem bereits bewährten Konzept mit folgenden Schwerpunkten ausgerichtet:
– Kommerzielle Hard- und Software aller marktführenden Hersteller für Handel, Handwerk, Industrie und Dienstleistung!
– Ständige Vorführungen von Standard-Software, wie Finanzbuchhaltung, Lohn- und Gehaltsabrechnung, Lager- und Auftragsverwaltung sowie der modernen Textverarbeitung für den Klein- und Mittelbetrieb!

– Besondere Programmprodukte für freiberuflich Tätige wie Rechtsanwälte, Ärzte, Steuerberater, Architekten usw.!
– Fachvorträge durch Referenten namhafter Firmen und dem Institut für Informatik in Saarbrücken!
– Große Auswahl an Fachliteratur!
– Audio- und Video sowie moderne Kommunikationstechniken!
– Umfangreicher Zubehörhandel vom Laserdrucker bis zur einfachen Diskette!
– Heimcomputer und Zubehör in allen Preisklassen für unsere jüngeren Messebesucher!
– Meßgeräte, Elektronikteile und Zubehör für Profis und Bastler!
– Besondere Vorführungen im Bereich Amateurfunk, wie Meteosatempfang, Funkfernschreiben per Computer usw., durch den Deutschen Amateur Radio Club, Ortsverband Völklingen.

Ein Super-Disassembler für den TI-99

Zum Programmieren in Maschinensprache benötigt man neben einem Assembler und einem Monitorprogramm auch einen Disassembler. Einen solchen stellt das folgende Programm dar, das um viele Besonderheiten erweitert wurde. Es läuft auf den TI-99 mit Extended Basic, Diskette und 32K-Speichererweiterung.
Disassembler werden verwendet, um z. B. das Betriebssystem eines Computers zu analysieren oder Fehler in eigenen Maschinenprogrammen aufzuspüren. Der abgedruckte Disassembler hat aber einige Zusatzfunktionen, die meist fehlen, obwohl diese sehr wichtig sind:
– Relative Sprünge und CRU-Bit-Befehle werden auch für negative Werte korrekt disassembliert, z. B. JOC $-8 oder TB -20. Dabei wird bei Relativsprüngen die absolute Adresse des Sprungziels berechnet.
– Der Pseudo-Befehl NOP (No Operation) wird erkannt.
– Die Befehle IDLE, RSET, CKON, CKOF und LREX, die zwar der TMS9900 versteht, die aber auf dem TI-99 nicht verwendet werden, werden ebenfalls disassembliert und mit einem Fragezeichen angezeigt, z. B. "IDLE?".
– Die wichtigste Besonderheit: Adressen im CPU-RAM von ›8300 bis ›83FF sowie der Sprung in den GPL-Interpreter NEXT (›0070) werden statt eines Zahlenwertes als Labels ausgegeben. Z. B. ist der Befehl SB §STATUS-.§JOYSTX erheblich leichter zu verstehen als SB §›837C,§›8377.
Zur Programmbedienung: Sobald es mit RUN gestartet wird, erscheint eine übersichtliche Eingabemaske. Hier geben Sie die Start- und die Endadresse des zu disassemblierenden Bereichs ein. Diese Eingaben können Sie dezimal oder hexadezimal machen. Im letzteren Fall müssen Sie das Hex-Symbol "›" voranstellen. Außerdem können Sie wählen, ob das Listing zusätzlich auf dem Drucker oder auf Diskette ausgegeben werden soll.

Kurz darauf beginnt das Programm mit dem Disassemblieren. Die Ausgaben haben auf dem Drucker und dem Bildschirm das gleiche Format, nur daß auf dem Drucker mehr Zeichen in jeder Zeile darstellbar sind.

```
0AA8  02E0  LWPI GPLWS
      83E0
0AAC  B80E  AB   R14,§TIMER
      8379
0AB0  C320  .MOV §UDFINT,R12
      83C4
0AB4  1301  JEQ  ›0AB8
0AB6  069C  BL   *R12
0AB8  04C8  CLR  R8
0ABA  02E0  LWPI INTWS
      83C0
0ABE  0380  RTWP
```

1. Spalte: Speicheradresse
2. Spalte: Inhalt der Adresse
3. Spalte: Assembler-Befehl
4. Spalte: Zugehörige Operanden

Folgende Tabellen werden statt der entsprechenden Adressen angezeigt:

›0070 NEXT = Sprung in den GPL-Interpreter
›2008 NUMASG = Numerischen Parameter zuweisen
›200C NUMREF = Numerischen Parameter holen
›2010 STRASG = Stringparameter zuweisen
›2014 STRREF = Stringparameter holen
›2018 XMLLNK = ROM-Routinen aufrufen
›201C KSCAN = Tastaturabfrage
›2020 VSBW = Ein Byte in den Video-RAM schreiben
›2024 VMBW = Mehrere Bytes in den Video-RAM schreiben
›2028 VSBR = Ein Byte aus dem Video-RAM lesen
›202C VMBR = Mehrere Bytes aus dem Video-RAM lesen
›2030 VMTR = VDP-Register schreiben
›2034 ERR = Fehlermeldung ausgeben
›8300 PAD = CPU-RAM-Pad
›831E BASCMD = Zeiger auf gerade bearbeiteten Basic-Befehl
›8322 ERRCOD = Fehlercode bei Rückkehr aus Maschinenprogramm
›832E ACTLIN = Zeiger auf gerade bearbeitete Basic-Zeile
›8330 LSTLIN = Zeiger auf Ende der Zeilenliste
›8332 FSTLIN = Zeiger auf Beginn der Zeilenliste
›833A SUBLST = Zeiger auf Liste der CALL-Unterprogramme
›833C PABLST = Zeiger auf Liste der PAB-Blöcke
›833E VARLST = Zeiger auf Liste der Variablen
›8340 VDPSTR = Zeiger auf freien Video-RAM für Strings
›8344 RUNFLG = Basic-Run-Flag (0 oder 255)
›8345 BASFLG = Basic-Flag-Byte (TRACE, PROTECTION, NUM etc.)
›834A FAC = Fließkomma-Akku 1
›8354 OVLERR = Fehlerflag bei Fließkomma-Operationen
›835C ARG = Fließkomma-Akku 2
›836E VSPTR = Value Stack Pointer
›8370 VHIMEM = Höchste nutzbare Adresse im Video-RAM
›8372 DATSTK = Zeiger auf GPL-Daten-Stack
›8373 SUBSTK = Zeiger auf GPL-Unterprogramm-Stack
›8374 KBOARD = In CALL KEY gewählter Tastaturmodus
›8375 ASCII = Wert der gedrückten Taste bei SCAN/KSCAN
›8376 JOYSTY = Y-Koordinate des Joysticks
›8377 JOYSTX = X-Koordinate des Joysticks
›8378 RANDOM = Zufallszahlen
›8379 INTCNT = Interruptgesteuerter Counter
›837A SPRMOT = Höchste Nummer der Sprites, die bewegt werden können
›837B VSTAT = VDP-Status-Register
›837C STATUS = GPL-Statusbyte
›8384 XHIMEM = Höchste nutzbare Adresse der Memory Expansion
›8386 XHFREE = Zeiger auf höchste freie Adresse der Memory Expansion
›83C0 INTWS = Interrupt-Workspace
›83C2 IFLAG = Interrupt-Flag
›83C4 UDFINT = User-defined Interrupt
›83CC SNDLST = Zeiger auf Soundliste für automatische Tonerzeugung
›83CE SNDBYT = Gerade gespieltes Soundbyte
›83D0 GRMSRC = GROM-Suchadresse
›83D2 ROMSRC = ROM-Suchadresse
›83D4 V1COPY = Kopie des VDP-Registers 1
›83D6 CLSCNT = Counter für Clearscreen
›83D8 SCANRT = Rücksprungadresse von SCAN/KSCAN
›83E0 GPLWS = GPL-Workspace

```
100 !  ********************
110 ! **********************
120 ! *                    *
130 ! *   DISASSEMBLER     *
140 ! *                    *
150 ! **********************
160 !  ********************
170 !
180 ! Für den TI-99 mit Extended Basic
190 !
200 ! (c) Martin Kotulla 1985
210 !
220 ! ------------------------------------------------------
230 !
240 DATA "SZC ","SZCB","S   ","SB  ","C   ","CB  "
250 DATA "A   ","AB  ","MOV ","MOVB","SOC ","SOCB"
260 DATA "JMP ","JLT ","JLE ","JEQ ","JHE ","JGT "
270 DATA "JNE ","JNC ","JOC ","JNO ","JL  ","JH  "
280 DATA "JOP ","SBO ","SBZ ","TB  "
290 DATA "COC ","CZC ","XOR "
300 DATA "LDCR","STCR"
```

```
310 DATA "SRA ","SRL ","SLA ","SRC "
320 DATA "BLWP","B   ","X   ","CLR ","NEG ","INV "
330 DATA "INC ","INCT","DEC ","DECT","BL  ","SWPB"
340 DATA "SETO","ABS "
350 DATA "LI  ","AI  ","ANDI","ORI ","CI  "
360 ! Labels für CPU-RAM im Bereich >8300 bis >83FF ******************************
370 DATA 0,PAD,30,BASCMD,34,ERRCOD,46,ACTLIN,48,LSTLIN,50,FSTLIN,58,SUBLST
380 DATA 60,PABLST,62,VARLST,64,VDPSTR,68,RUNFLG,69,BASFLG,74,FAC
390 DATA 84,OVLERR,92,ARG,110,VSPTR,112,VHIMEM,114,DATSTK,115,SUBSTK
400 DATA 116,KBOARD,117,ASCII,118,JOYSTY,119,JOYSTX,120,RANDOM,121,INTCNT
410 DATA 122,SPRMOT,123,VSTAT,124,STATUS,132,XHIMEM,134,XHFREE,192,INTWS
420 DATA 194,IFLAG,196,UDFINT,204,SNDLST,206,SNDBYT,208,GRMSRC,210,ROMSRC
430 DATA 212,V1COPY,214,CLSCNT,216,SCANRT,224,GPLWS
440 ! Labels für Assembler-Hilfsprogramme im Low Memory von >2000 - >24F4 ******
450 DATA NUMASG,NUMREF,STRASG,STRREF,XMLLNK,KSCAN,VSBW,VMBW,VSBR,VMBR,VWTR,ERR
460 ! Definitionen und Daten einlesen ***********************************************
470 DIM FM1$(11),FM2$(15),FM3$(2),FM4$(1),FM5$(3)
480 DIM FM6$(13),FM8$(4),LABEL$(1,255)
490 FOR I=0 TO 11 :: READ FM1$(I):: NEXT I
500 FOR I=0 TO 15 :: READ FM2$(I):: NEXT I
510 READ FM3$(0),FM3$(1),FM3$(2)
520 READ FM4$(0),FM4$(1)
530 READ FM5$(0),FM5$(1),FM5$(2),FM5$(3)
540 FOR I=0 TO 13 :: READ FM6$(I):: NEXT I
550 FOR I=0 TO 4 :: READ FM8$(I):: NEXT I
560 HX$="0123456789ABCDEF"
570 DEF HEX$(X)=SEG$(HX$,INT(X/16)+1,1)&SEG$(HX$,X-INT(X/16)*16+1,1)
580 DEF HEXW$(X)=HEX$(INT(X/256))&HEX$(X-INT(X/256)*256)
590 DEF DEZ(X$)=(POS(HX$,SEG$(X$,1,1),1)-1)*16+POS(HX$,SEG$(X$,2,1),1)-1
600 DEF DEZW(X$)=DEZ(SEG$(X$,1,2))*256+DEZ(SEG$(X$,3,2))
610 DEF COMPL2=I+65536*(I)32767)! Zweierkomplement einer Integerzahl
620 ! Titelbild ***************************************************************************
630 CALL CHAR(42,"AA55AA55AA55AA55",91,"7E8199A1A199817E")
640 CALL CHAR(136,RPT$("0",8)&"FF")
650 CALL CHAR(128,"DFBF60C0919B959191808080C060BFDF")
660 CALL CHAR(130,"FBFD06030101014951615149030FDFB")
670 CALL CLEAR :: CALL HCHAR(2,4,42,26)
680 CALL VCHAR(3,4,42,5):: CALL VCHAR(3,29,42,5)
690 CALL MAGNIFY(3):: CALL SPRITE(#1,128,2,28,60)
700 DISPLAY AT(5,11)SIZE(12):"DISASSEMBLER"
710 CALL HCHAR(8,4,42,26)
720 DISPLAY AT(12,4):CHR$(91)&"  MARTIN KOTULLA 1985"
730 CALL HCHAR(13,6,136,22)
740 DISPLAY AT(15,1):"- STARTADRESSE?": :"- ENDADRESSE?"
750 DISPLAY AT(19,1):"- DRUCKERAUSGABE?  N"
760 ACCEPT AT(15,20)SIZE(5)VALIDATE(HX$,")"):ST$
770 IF ASC(ST$)=62 THEN START=DEZW(SEG$(ST$,2,4))ELSE START=VAL(ST$)
780 START=INT(START/2)*2
790 ACCEPT AT(17,20)SIZE(5)VALIDATE(HX$,")"):EN$
800 IF ASC(EN$)=62 THEN ENDE=DEZW(SEG$(EN$,2,4))ELSE ENDE=VAL(EN$)
810 ENDE=INT(ENDE/2)*2
820 ACCEPT AT(19,20)SIZE(-1)VALIDATE("JN"):FILE$
830 IF FILE=1 THEN CLOSE #1 :: FILE=0
840 IF FILE$="N" THEN FILE=0 :: GOTO 890
850 FILE=1
860 DISPLAY AT(21,1):"- DRUCKERNAME? RS232"
870 ACCEPT AT(21,16)SIZE(-12)VALIDATE(UALPHA,DIGIT,".="):DRUCK$
880 OPEN #1:DRUCK$
890 CALL CLEAR :: CALL CHARSET :: CALL DELSPRITE(#1)
900 FOR I=START TO ENDE STEP 2
910 CALL PEEK(COMPL2,HIBYTE,LOBYTE)
920 WORD=HIBYTE*256+LOBYTE
930 IF WORD<32768 THEN NEGWORD=WORD ELSE NEGWORD=WORD-65536
940 OUT$=HEXW$(I)&" "&HEXW$(WORD)&" "
950 STARTCMD=I
```

```
960 ! Pseudo-Befehl NOP ***
970 IF WORD=4096 THEN OUT$=OUT$&"NOP" :: GOTO 1270
980 ! Format 1 **************
990 IF WORD>16383 THEN 1480
1000 ! Format 2a ************
1010 IF WORD>4095 AND WORD<7424 THEN 1570
1020 ! Format 2b ************
1030 IF WORD>7423 AND WORD<8192 THEN 1640
1040 ! Format 3 **************
1050 IF WORD>8191 AND WORD<11264 THEN 1690
1060 ! Format 4 **************
1070 IF WORD>12287 AND WORD<14336 THEN 1770
1080 ! Format 5 **************
1090 IF WORD>2047 AND WORD<3072 THEN 1850
1100 ! Format 6 **************
1110 IF WORD>1023 AND WORD<2048 THEN 1910
1120 ! Format 7 **************
1130 IF WORD=832 OR WORD=864 OR WORD=896 THEN 1970
1140 IF WORD=928 OR WORD=960 OR WORD=992 THEN 1970
1150 ! Format 8a ************
1160 IF WORD>511 AND WORD<672 THEN 2050
1170 ! Format 8b ************
1180 IF WORD=736 OR WORD=768 THEN 2120
1190 ! Format 8c ************
1200 IF WORD>671 AND WORD<720 THEN 2160
1210 ! Format 9a ************
1220 IF WORD>11263 AND WORD<12288 THEN 2210
1230 ! Format 9b ************
1240 IF WORD>14335 AND WORD<16384 THEN 2280
1250 ! DATA-Befehle *********
1260 OUT$=OUT$&"DATA )"&HEXW$(WORD)
1270 CALL HCHAR(24,1,ASC(OUT$))
1280 CALL HCHAR(24,2,ASC(SEG$(OUT$,2,1)))
1290 IF LEN(OUT$)<31 THEN PRINT SEG$(OUT$,3,255):: GOTO 1360
1300 PRINT SEG$(OUT$,3,28)
1310 CALL HCHAR(23,31,ASC(SEG$(OUT$,31,1)&"    "))
1320 CALL HCHAR(23,32,ASC(SEG$(OUT$,32,1)&"    "))
1330 IF LEN(OUT$)<33 THEN 1360
1340 PRINT :: FOR K=33 TO LEN(OUT$)
1350 CALL HCHAR(23,K-17,ASC(SEG$(OUT$,K,1))):: NEXT K
1360 IF FILE=1 THEN PRINT #1:OUT$
1370 X1=STARTCMD+2 :: IF X1>32767 THEN X1=X1-65536
1380 X2=COMPL2
1390 FOR J=X1 TO X2 STEP 2
1400 CALL PEEK(J,A,B)
1410 PRINT TAB(4);HEXW$(A*256+B)
1420 IF FILE=1 THEN PRINT #1:TAB(6);HEXW$(A*256+B)
1430 NEXT J
1440 NEXT I
1450 PRINT :: DISPLAY BEEP:"FERTIG-BITTE SPACE DRUECKEN!"
1460 CALL KEY(0,K,S):: IF K<>32 THEN 1460 ELSE 630
1470 ! FORMAT 1 ***************************************************************************
1480 OUT$=OUT$&FM1$(INT(WORD/4096)-4)&" "
1490 DESTINATION=15 AND(INT(WORD/64))
1500 SOURCE=15 AND NEGWORD
1510 TYPEDEST=3 AND(INT(WORD/1024))
1520 TYPESOURCE=3 AND(INT(WORD/16))
1530 CALL DECODE(SOURCE,TYPESOURCE,I,"S",OUT$)! SUB "OPERANDEN DEKODIEREN"
1540 CALL DECODE(DESTINATION,TYPEDEST,I,"D",OUT$)! SUB "OPERANDEN DEKODIEREN"
1550 GOTO 1270
1560 ! FORMAT 2a ***************************************************************************
1570 OUT$=OUT$&FM2$(INT(WORD/256)-16)&" "
1580 DISPLACE=LOBYTE*2
1590 IF DISPLACE>255 THEN DISPLACE=DISPLACE-512
```

```
1600 JUMPADDR=I+DISPLACE+2
1610 IF JUMPADDR=112 THEN OUT$=OUT$&"NEXT" :: GOTO 1270
1620 OUT$=OUT$&")"&HEXW$(JUMPADDR):: GOTO 1270
1630 ! FORMAT 2b ***************************************************************
1640 OUT$=OUT$&FM2$(INT(WORD/256)-16)&" "
1650 WORD=WORD AND 255
1660 IF WORD<128 THEN OUT$=OUT$&STR$(WORD)ELSE OUT$=OUT$&STR$(WORD-256)
1670 GOTO 1270
1680 ! FORMAT 3 ***************************************************************
1690 OUT$=OUT$&FM3$(INT(WORD/1024)-8)&" "
1700 TYPESOURCE=INT(WORD/16)AND 3
1710 SOURCE=NEGWORD AND 15
1720 CALL DECODE(SOURCE,TYPESOURCE,I,"S",OUT$)! SUB "OPERANDEN DEKODIEREN"
1730 DESTINATION=INT(WORD/64)AND 15
1740 OUT$=OUT$&"R"&STR$(DESTINATION)
1750 GOTO 1270
1760 ! FORMAT 4 ***************************************************************
1770 OUT$=OUT$&FM4$(INT(WORD/1024)-12)&" "
1780 TYPESOURCE=INT(WORD/16)AND 3
1790 SOURCE=NEGWORD AND 15
1800 CALL DECODE(SOURCE,TYPESOURCE,I,"S",OUT$)! SUB "OPERANDEN DEKODIEREN"
1810 CRULINE=INT(WORD/64)AND 15
1820 OUT$=OUT$&STR$(CRULINE)
1830 GOTO 1270
1840 ! FORMAT 5 ***************************************************************
1850 OUT$=OUT$&FM5$(HIBYTE-8)&" "
1860 REGISTER=LOBYTE AND 15
1870 SHIFTCOUNT=INT(LOBYTE/16)AND 15
1880 OUT$=OUT$&"R"&STR$(REGISTER)&","&STR$(SHIFTCOUNT)
1890 GOTO 1270
1900 ! FORMAT 6 ***************************************************************
1910 OUT$=OUT$&FM6$(INT(WORD/64)-16)&" "
1920 TYPESOURCE=INT(WORD/16)AND 3
1930 SOURCE=WORD AND 15
1940 CALL DECODE(SOURCE,TYPESOURCE,I,"D",OUT$)! SUB "OPERANDEN DEKODIEREN"
1950 GOTO 1270
1960 ! FORMAT 7 ***************************************************************
1970 IF WORD=832 THEN OUT$=OUT$&"IDLE?"
1980 IF WORD=864 THEN OUT$=OUT$&"RSET?"
1990 IF WORD=896 THEN OUT$=OUT$&"RTWP"
2000 IF WORD=928 THEN OUT$=OUT$&"CKON?"
2010 IF WORD=960 THEN OUT$=OUT$&"CKOF?"
2020 IF WORD=992 THEN OUT$=OUT$&"LREX?"
2030 GOTO 1270
2040 ! FORMAT 8a ***************************************************************
2050 OUT$=OUT$&FM8$(INT(WORD/32)-16)&" "
2060 REGISTER=NEGWORD AND 15
2070 OUT$=OUT$&"R"&STR$(REGISTER)
2080 CALL DEEK(I,IMMEDIATE)
2090 OUT$=OUT$&",)"&HEXW$(IMMEDIATE)
2100 GOTO 1270
2110 ! FORMAT 8b ***************************************************************
2120 CALL DEEK(I,IMMEDIATE)
2130 IF WORD=736 THEN OUT$=OUT$&"LWPI )"&HEXW$(IMMEDIATE):: CALL LABEL(I,OUT$)::
 GOTO 1270
2140 OUT$=OUT$&"LIMI "&STR$(IMMEDIATE):: GOTO 1270
2150 ! FORMAT 8c ***************************************************************
2160 IF WORD<704 THEN OUT$=OUT$&"STWP" ELSE OUT$=OUT$&"STST"
2170 REGISTER=NEGWORD AND 15
2180 OUT$=OUT$&" R"&STR$(REGISTER)
2190 GOTO 1270
2200 ! FORMAT 9a ***************************************************************
2210 OUT$=OUT$&"XOP  "
2220 SOURCE=NEGWORD AND 15
2230 TYPESOURCE=INT(WORD/16)AND 3
```

```
2240 CALL DECODE(SOURCE,TYPESOURCE,I,"S",OUT$)! SUB "OPERANDEN DEKODIEREN"
2250 DESTINATION=INT(WORD/64)AND 15
2260 OUT$=OUT$&STR$(DESTINATION):: GOTO 1270
2270 ! FORMAT 9b ***********************************************************************
2280 IF WORD(15360 THEN OUT$=OUT$&"MPY  " ELSE OUT$=OUT$&"DIV  "
2290 SOURCE=NEGWORD AND 15
2300 TYPESOURCE=INT(WORD/16)AND 3
2310 CALL DECODE(SOURCE,TYPESOURCE,I,"S",OUT$)! SUB "OPERANDEN DEKODIEREN"
2320 DESTINATION=INT(WORD/64)AND 15
2330 OUT$=OUT$&"R"&STR$(DESTINATION):: GOTO 1270
2340 ! ***********************************************************************
2350 ! UNTERPROGRAMME ******************************************************
2360 ! ***********************************************************************
2370 ! Adressierungsart dekodieren ------------------------------------------
2380 SUB DECODE(OPERAND,TYPE,I,SD$,OUT$)
2390 DEF COMPL2=I+65536*(I)32767)
2400 ON TYPE+1 GOTO 2420,2440,2460,2550
2410 ! *-* WORKSPACE REGISTER *-*
2420 OUT$=OUT$&"R"&STR$(OPERAND):: GOTO 2560
2430 ! *-* WORKSPACE REGISTER INDIRECT *-*
2440 OUT$=OUT$&"*R"&STR$(OPERAND):: GOTO 2560
2450 ! *-* SYMBOLIC MEMORY/INDEXED MEMORY *-*
2460 I=I+2 :: CALL PEEK(COMPL2,A,B)
2470 HX$="0123456789ABCDEF"
2480 HX1$=SEG$(HX$,INT(A/16)+1,1)&SEG$(HX$,A-INT(A/16)*16+1,1)
2490 HX2$=SEG$(HX$,INT(B/16)+1,1)&SEG$(HX$,B-INT(B/16)*16+1,1)
2500 IF OPERAND<>0 THEN 2520
2510 OUT$=OUT$&CHR$(64)&")"&HX1$&HX2$ :: CALL LABEL(I,OUT$):: GOTO 2560
2520 OUT$=OUT$&CHR$(64)&")"&HX1$&HX2$ :: CALL LABEL(I,OUT$)
2530 OUT$=OUT$&"(R"&STR$(OPERAND)&")" :: GOTO 2560
2540 ! *-* WORKSPACE REGISTER INDIRECT AUTO-INCREMENT *-*
2550 OUT$=OUT$&"*R"&STR$(OPERAND)&"+"
2560 IF SD$="S" THEN OUT$=OUT$&","
2570 SUBEND
2580 ! Double Peek (16-Bit-Wort lesen) ----------------------------------------
2590 SUB DEEK(I,C)
2600 I=I+2 :: COMPL2=I+65536*(I)32767)
2610 CALL PEEK(COMPL2,A,B)
2620 C=A*256+B :: SUBEND
2630 ! Adresse durch Label ersetzen ---------------------------------------------
2640 SUB LABEL(I,OUT$)
2650 IF FLG THEN 2690 ELSE FLG=-1
2660 RESTORE 370 :: DIM LABEL$(1,255)
2670 FOR J=0 TO 40 :: READ ADR,LABEL$(0,ADR):: NEXT J
2680 FOR J=0 TO 11 :: ADR=J*4+8 :: READ LABEL$(1,ADR):: NEXT J
2690 CALL PEEK(I,A,B)
2700 IF A*256+B=112 THEN OUT$=SEG$(OUT$,1,LEN(OUT$)-5)&"NEXT" :: SUBEXIT
2710 IF A<>131 THEN 2740
2720 IF LABEL$(0,B)<>"" THEN OUT$=SEG$(OUT$,1,LEN(OUT$)-5)&LABEL$(0,B)
2730 SUBEXIT
2740 IF A<>32 THEN 2760
2750 IF LABEL$(1,B)<>"" THEN OUT$=SEG$(OUT$,1,LEN(OUT$)-5)&LABEL$(1,B)
2760 SUBEND
2770 END ! ************************************************************************
```

Zeichen-Definition in TI-BASIC für die Grundversion

Die drei großen Vorteile des TI-Programms

- „Zeichen-Definition" ist wirklich eine große Hilfe für alle TI-Basic-Programmierer, denn es erspart das umständliche und zeitaufwendige Erstellen von Zeichencodes im Hexadezimal-System. Diese Codes werden gebraucht, um ein selbsterstelltes Zeichen mit „CALL CHAR" zu definieren.
- „Zeichen-Definition" läuft in TI-Basic.
  Für fast alle CPU-UTILITY-Programme für den TI99/4A wurde entweder das Extended-Basic-Modul, eine Speichererweiterung oder gar ein Disketten-Laufwerk gebraucht. Aber was ist mit den zahllosen TI-Usern, die nur die Grundversion besitzen? Ein Großteil der TI-Besitzer kann sich halt keine Erweiterung, kein Disketten-Laufwerk leisten.
  Für all diese TI-User ist „Zeichen-Definition" ideal: Es läuft tatsächlich nur auf der Konsole! Es werden weder irgendwelche Erweiterungen oder Module, noch Joysticks oder anderes Zubehör benötigt. Nur die einfachste Grundversion reicht, um „Zeichen-Definition" zu benutzen.
- „Zeichen-Definition" ist überaus einfach zu bedienen.
  Durch konsequente Anwendung der bekannten Menue-Technik kann das Programm wirklich von jedermann bedient werden, der den TI von einer Waschmaschine unterscheiden kann. Alle Tasten, die benutzt werden können, werden auf dem Bildschirm genannt.

Noch einige weitere Pluspunkte:

- Das Programm ist durch REM-Zeilen ausführlich kommentiert. Jeder TI-User, der selbst programmiert, wird sich in dem Programm zurechtfinden. Es kann jederzeit geändert werden, um es auf individuelle Bedürfnisse zuzuschneiden.
- Sollte man beim Entwerfen des Zeichen-Musters einen Fehler machen, so kann man diesen durch einen einzigen Tastendruck wieder löschen.
- Die benötigten Tasten sind nach ergonomischen Gesichtspunkten bestmöglich gewählt: Mit der linken Hand bewegt man mit den Pfeil-Tasten den blauen Cursor, mit der rechten Hand legt man einen Punkt fest (Enter-Taste) oder man löscht einen gesetzten Punkt („="-Taste). Ist man mit dem Entwerfen fertig und möchte den Zeichencode definiert haben, so drückt man nur die „1"-Taste. Diese ist gewählt worden, da sie am weitesten von den anderen benutzten Tasten entfernt ist; man kann sie nicht aus Versehen bedienen.
- Der Hexadezimal-Code, den man ja meistens braucht, um in einem eigenen Programm mit „Call Char" ein Zeichen zu definieren, wird nicht nur auf dem Bildschirm angezeigt (man kann ihn nun abschreiben), sondern er kann auf Wunsch auch abgedruckt werden.
- Um sich das entworfene Zeichen in Originalgröße anzuschauen, besteht auch die Möglichkeit der „Call Char"-Definition im Programm selbst.

Zum Laden des Programms:
Nach der Eingabe des Programms und dem Befehl „RUN" erscheint ein Raster aus 8 mal 8 leeren Feldern. Dies würde einem Zeichen entsprechen, bei dem keiner der 64 Punkte gesetzt ist (Leerzeichen). Würde man alle Felder schwarz füllen, so entstände ein Zeichen, das einem einfarbigen Block entsprechen würde.
Das blaue Viereck ist der Cursor. Der Cursor wird mit Hilfe der vier Pfeil-Tasten über das Raster bewegt (E, S, D und X).
Möchte man nun einen Punkt festlegen, so muß man die ENTER-Taste drücken. Das Feld färbt sich dann schwarz, der Punkt ist gesetzt. Möchte man den Punkt wieder löschen, so muß man die „="-Taste betätigen.
Sobald man mit dem Entwurf des gewünschten Musters fertig ist, muß man schließlich die Taste „1" drücken, damit der Computer den Hexadezimal-Code berechnet. Diesen Vorgang kann man durch die Anzeige des Binärcodes verfolgen (0 = nicht gesetzt; 1 = gesetzt).
Sobald der Hexadezimal-Code angezeigt wird, sollte man sich diesen aufschreiben, falls man ihn in einem eigenen Programm verwenden möchte, um ein Zeichen mit „CALL CHAR" zu definieren.
Durch Druck auf die Taste „1" kann der Code auch auf dem Thermoprinter ausgedruckt werden. Besitzt man einen anderen Drucker, so sollte man die Zeilen 2100 bis 2120 abändern. Hat man keinen Drucker, so sollte man diese Zeilen mit REM überschreiben, um eine Fehlermeldung bei versehentlicher Betätigung der Taste „1" zu vermeiden.
Drückt man die Taste „2", so kann man sich das definierte Zeichen in Originalgröße anschauen. Es wird in diesem Fall mit CALL CHAR definiert. Den Zeichencode (von 144 bis 159) kann man selbst wählen.
Auf Druck der Taste „3" beginnt das Programm von Anfang an neu, um ein anderes Zeichen zu definieren, und auf Druck der Taste „4" endet das Programm.

Programm-Aufbau:

100– 370	Vorbereitungen (Variablen, Farben-, Zeichen-Definition etc.)
340–1120	Haupt-Screen (Cursor bewegen, Punkte ,setzen und löschen etc.)
1130–1940	Zeichencode berechnen (Binärcode anzeigen, Hex.-Code anzeigen)
1950–2270	Möglichkeiten 1–4 (Code drukken, Zeichen mit CALL CHAR definieren, von Anfang an neu beginnen, Ende)

An allen sinnvollen Stellen wurden Sounds verwendet, um das Programm interessanter zu gestalten. Das Programm läuft auch in Extended Basic, wenn man wenige Kleinigkeiten abändert (Zeichen-, Farb-Codes) und ist nicht zu lang, um es auf Diskette abzuspeichern. Es ist jedoch, wie bereits gesagt, durchaus auf der „nackten" TI99-Konsole in TI-Basic lauffähig.

Das Listing für dieses Programm finden Sie auf Seite 10

```
100 REM
110 REM ** ZEICHEN DEFINIEREN          ** TI99-4A (TI BASIC)          ** (C)1985
120 REM ** PATRICK SCHMITZ             ** AM ZEHNTHOF 4               ** 548 REM.-UNKE
LBACH
130 REM
140 REM  ** VORBEREITUNGEN
150 REM
160 Z=0
170 Z4=1
180 Z5=1
190 Z6=0
200 CHAR$=""
210 ZCHAR$=""
220 CALL COLOR(13,16,1)
230 CALL COLOR(14,5,1)
240 CALL CLEAR
250 CALL CHAR(128,"FF8181818181FF")
260 CALL CHAR(136,"FFFFFFFFFFFFFFFF")
270 CALL CHAR(144,"FFFFFFFFFFFFFFFF")
280 ZEILE=8
290 @ZEILE=8
300 SPALTE=11
310 @SPALTE=11
320 DIM GZEILE(200)
330 DIM GSPALTE(200)
340 REM
350 REM  ** HAUPTSCREEN
360 REM
370 PRINT " *** ZEICHENDEFINITION *** "::::::::::::::::::::::::::
380 PRINT "* CURSOR-BEWEGUNG: ""E,S,D,X""":"* PUNKT FESTLEGEN: "" ENTER """
390 PRINT "* PUNKT LOESCHEN : ""="" TASTE":"* ZEICHEN DEFIN.  : ""1"" TASTE"::::
::::::::::::::::
400 FOR A=8 TO 15
410 FOR B=11 TO 18
420 CALL HCHAR(A,B,128)
430 NEXT B
440 NEXT A
450 CALL HCHAR(8,11,136)
460 REM
470 REM  ** TASTEN-ABFRAGE
480 REM
490 CALL KEY(3,XX,YY)
500 IF YY=0 THEN 490
510 IF YY=-1 THEN 490
520 IF XX=13 THEN 900
530 IF XX=68 THEN 620
540 IF XX=69 THEN 650
550 IF XX=83 THEN 680
560 IF XX=61 THEN 1030
570 IF XX=49 THEN 1160
580 IF XX=88 THEN 710 ELSE 490
590 REM
600 REM  ** CURSOR-BEWEGUNG
610 REM
620 IF SPALTE>17 THEN 490
630 LET SPALTE=SPALTE+1
640 GOTO 730
650 IF ZEILE<9 THEN 490
660 ZEILE=ZEILE-1
670 GOTO 730
680 IF SPALTE<12 THEN 490
690 SPALTE=SPALTE-1
700 GOTO 730
710 IF ZEILE>14 THEN 490
720 LET ZEILE=ZEILE+1
730 CALL SOUND(4000,750,5)
740 FOR Z1=1 TO Z
750 IF @SPALTE=GSPALTE(Z1)THEN 760 ELSE 770
760 IF @ZEILE=GZEILE(Z1)THEN 790
770 NEXT Z1
780 GOTO 810
790 CALL HCHAR(@ZEILE,@SPALTE,144)
800 GOTO 820
810 CALL HCHAR(@ZEILE,@SPALTE,128)
820 CALL HCHAR(ZEILE,SPALTE,136)
830 @ZEILE=ZEILE
840 @SPALTE=SPALTE
850 CALL SOUND(-10,500,1)
860 GOTO 490
870 REM
880 REM  ** PUNKT SETZEN
890 REM
900 CALL SOUND(100,1000,0)
910 FOR Z0=1 TO Z
920 IF GZEILE(Z0)<>ZEILE THEN 940 ELSE 930
930 IF GSPALTE(Z0)<>SPALTE THEN 940 ELSE 990
940 NEXT Z0
950 Z=Z+1
960 LET GZEILE(Z)=ZEILE
970 LET GSPALTE(Z)=SPALTE
980 CALL HCHAR(ZEILE,SPALTE,144)
990 GOTO 490
1000 REM
1010 REM  ** PUNKT LOESCHEN
1020 REM
1030 CALL SOUND(100,110,0)
1040 FOR Z2=1 TO Z
1050 IF ZEILE<>GZEILE(Z2)THEN 1070 ELSE 1060
```

```
1060 IF SPALTE<>GSPALTE(Z2)THEN 1070 ELSE 1090
1070 NEXT Z2
1080 GOTO 490
1090 LET GZEILE(Z2)=0
1100 LET GSPALTE(Z2)=0
1110 CALL HCHAR(ZEILE,SPALTE,128)
1120 GOTO 490
1130 REM
1140 CALL SOUND(100,800,2)
1150 CALL SOUND(100,600,2)
1160 CALL SOUND(100,700,3)
1170 Z4=1
1180 Z5=1
1190 PRINT "           BINAER:"
1200 FOR Z3=1 TO Z
1210 IF GZEILE(Z3)-7<>Z5 THEN 1240
1220 IF GSPALTE(Z3)-10<>Z4 THEN 1240ELSE 1260
1230 NEXT Z3
1240 BIT=0
1250 GOTO 1270
1260 BIT=1
1270 CHAR$=CHAR$&STR$(BIT)
1280 Z4=Z4+1
1290 IF LEN(CHAR$)<4 THEN 1200
1300 IF CHAR$<>"0000" THEN 1310 ELSE 1480
1310 IF CHAR$<>"0001" THEN 1320 ELSE 1500
1320 IF CHAR$<>"0010" THEN 1330 ELSE 1520
1330 IF CHAR$<>"0011" THEN 1340 ELSE 1540
1340 IF CHAR$<>"0100" THEN 1350 ELSE 1560
1350 IF CHAR$<>"0101" THEN 1360 ELSE 1580
1360 IF CHAR$<>"0110" THEN 1370 ELSE 1600
1370 IF CHAR$<>"0111" THEN 1380 ELSE 1620
1380 IF CHAR$<>"1000" THEN 1390 ELSE 1640
1390 IF CHAR$<>"1001" THEN 1400 ELSE 1660
1400 IF CHAR$<>"1010" THEN 1410 ELSE 1680
1410 IF CHAR$<>"1011" THEN 1420 ELSE 1700
1420 IF CHAR$<>"1100" THEN 1430 ELSE 1720
1430 IF CHAR$<>"1101" THEN 1440 ELSE 1740
1440 IF CHAR$<>"1110" THEN 1450 ELSE 1760
1450 IF CHAR$<>"1111" THEN 1460 ELSE 1780
1460 CALL SOUND(150,-3,0)
1470 GOTO 490
1480 ZCHAR$=ZCHAR$&"0"
1490 GOTO 1840
1500 ZCHAR$=ZCHAR$&"1"
1510 GOTO 1840
1520 ZCHAR$=ZCHAR$&"2"
1530 GOTO 1840
1540 ZCHAR$=ZCHAR$&"3"
1550 GOTO 1840
1560 ZCHAR$=ZCHAR$&"4"
1570 GOTO 1840
1580 ZCHAR$=ZCHAR$&"5"
1590 GOTO 1840
1600 ZCHAR$=ZCHAR$&"6"
1610 GOTO 1840
1620 ZCHAR$=ZCHAR$&"7"
1630 GOTO 1840
1640 ZCHAR$=ZCHAR$&"8"
1650 GOTO 1840
1660 ZCHAR$=ZCHAR$&"9"
1670 GOTO 1840
1680 ZCHAR$=ZCHAR$&"A"
1690 GOTO 1840
1700 ZCHAR$=ZCHAR$&"B"
1710 GOTO 1840
1720 ZCHAR$=ZCHAR$&"C"
1730 GOTO 1840
1740 ZCHAR$=ZCHAR$&"D"
1750 GOTO 1840
1760 ZCHAR$=ZCHAR$&"E"
1770 GOTO 1840
1780 ZCHAR$=ZCHAR$&"F"
1790 GOTO 1840
1800 CALL SOUND(100,-3,0)
1810 GOTO 490
1820 Z6=Z6+1
1830 IF Z6=2 THEN 1860 ELSE 1900
1840 Z6=0
1850 Z4=1
1860 Z5=Z5+1
1870 GOTO 1910
1880 Z4=5
1890 PRINT " ";CHAR$;"  ";
1900 CHAR$=""
1910 IF Z5<=8 THEN 1200 ELSE 1920
1920 PRINT :"HEXADEZIMAL:";ZCHAR$
1930 REM
1940 REM  ** MENUE
1950 REM
1960 PRINT :"-1- HEXADEZIMALCODE DRUCKEN":"-2- ZEICHEN MIT CALL CHAR "
1970 PRINT "    DEFINIEREN"
1980 PRINT "-3- NEUES ZEICHEN DEFINIEREN":"-4- PROGRAMM BEENDEN"::
1990 CALL KEY(0,X,Y)
2000 IF Y=0 THEN 1990
2010 IF X=49 THEN 2100
2020 IF X=50 THEN 2170
2030 IF X=51 THEN 160
2040 IF X=52 THEN 2270 ELSE 1990
2050 REM
2060 REM * HEX.CODE DRUCKEN
2070 REM
2080 OPEN #1:"TP",OUTPUT
2090 PRINT #1:"MUSTER-KODIERUNG: ";ZCHAR$
2100 CLOSE #1
2110 GOTO 1960
2120 REM
2130 REM ** CALL CHAR DEF.
```

```
2160 REM
2170 INPUT "WELCHER ZEICHENCODE(144-159)?";ZEICHEN
2180 IF ZEICHEN<144 THEN 2170
2190 IF ZEICHEN>159 THEN 2170
2200 IF ABS(INT(ZEICHEN))<>ZEICHEN THEN 2170
2210 CALL CHAR(ZEICHEN,ZCHAR$)
2220 PRINT TAB(14);CHR$(ZEICHEN)
2230 GOTO 1980
2240 REM
2250 REM  ** PROGRAMM-ENDE
2260 REM
2270 END
```

Anbieter wollen den Weltstandard für kompatible Home-Computer gemeinsam populär machen

Die MSX-Arbeitsgemeinschaft Deutschland, gegründet im Rahmen der Hannover-Messe, besteht aus zunächst neun namhaften Herstellern und Vertriebsfirmen, die mit dem freiwilligen Zusammenschluß ihre Entschlossenheit dokumentieren, dem neuen kompatiblen Weltstandard für Computer im Heim-, Hobby-, Schul-, Spiel- und Profibereich gemeinsam auch in Deutschland zum Durchbruch zu verhelfen.

Die Gründung der MSX-Arbeitsgemeinschaft Deutschland soll als Signal verstanden werden: Erstmals schalten sich Firmen ins Computer-Geschäft ein, die bereits über meist jahrzehntelange Erfahrung in der Unterhaltungs- und Gebrauchselektronik verfügen. Voraussetzung für ein solches Engagement der großen Anbieter war die Schaffung eines einheitlichen Weltstandards für Betriebssysteme, Software und Peripherieschnittstellen. Erst eine solche Einigung eröffnet die Chance, den Computer für Privathaushalt und Büro in ausreichenden Stückzahlen abzusetzen.

Durch solchen Wettbewerb ist sichergestellt, daß die Wünsche und Bedürfnisse des Verbrauchers mit einem breiten Angebot an zwar grundsätzlich kompatibler, in Ausstattung und Preis jedoch höchst unterschiedlicher Hardware und Software zur Verfügung steht.

Erste Maßnahmen der neugegründeten MSX-Arbeitsgemeinschaft war die Schaffung eines gemeinsamen „MSX-Kontaktbüros".

In Fragen der individuellen Geräte-Ausstattung stehen nach wie vor die Presse-Sprecher der beteiligten Unternehmen zur Verfügung.

Weiterhin wird die MSX-Arbeitsgemeinschaft zu einem frühestmöglichen Zeitpunkt namhafte Software-Häuser zu einem Symposium laden, um auf Fragen zur Systemtechnik und zum Standard selbst kompetente Anwort zu geben.

Computerclub RUN

Unser Computerclub RUN sucht noch Mitglieder. Es erscheint monatlich eine Clubzeitung in der Anwender- bzw. Spiele-Listings, Tips und Tricks, Flugsimmulatoren und vieles mehr erscheinen. Außerdem ist ein Sonderteil vorhanden, der sich mit der Datenfernübertragung beschäftigt. Wir haben aber auch einen Software-Service in den man super-billig Supergams kaufen kann. Außerdem werden auch Listings von Spielen abgedruckt, die zur Zeit auf dem deutschen bzw. englischen Softwaremarkt der absolute Renner sind. Und dabei kommt keiner der bei uns behandelten Computer zu kurz. Alle werden gleich behandelt bzw. beschrieben, ob nun kleiner Speicher oder großer Speicher, teuer oder billig, alt oder neu. Jeder kann, wenn er will, in unserem Club bzw. unserer Clubzeitung mitmachen. Es werden sogar gute Honorare gezahlt. Außerdem: regelmäßig Preisausschreiben. Kleinanzeigen sind kostenlos!!! Also ran an die Feder (oder Drucker) und Infoblätter bzw. Anmeldeformulare bestellen. Es lohnt sich bestimmt, da ja alles umsonst ist! Wir beschäftigen uns hauptsächlich mit den COMODORE-Computern: VC-20, C-64, C-16, C-128. Aber auch Ataris und der Schneider CPC werden behandelt.

Infoblätter und Anmeldeformulare von:

Ralf Klinke
Unter Langscheid 6
5800 Hagen 8

SUPERFILE FÜR DEN TI-99

Laden und Saven mit erhöhter Geschwindigkeit

Wohl jeder TI-Benutzer, der den Cassettenrecorder zur Datenspeicherung verwenden wollte, dürfte vom Computer zum Wahnsinn getrieben worden sein. Geschwindigkeit und Datensicherheit sind so schlecht, daß nur die Anschaffung einer Diskettenstation übrigblieb. Das Programm SUPERFILE macht diesen Problemen ein Ende, es speichert die Daten mit dem Vielfachen der normalen Geschwindigkeit und sogar noch erhöhter Datensicherheit ab. Sie benötigen dafür einen TI-99 mit Extended Basic und 32K-Speichererweiterung.

Zuerst ein kleiner Exkurs in das Betriebssystem des TI-99: Sollen Daten auf irgendeinem Peripheriegerät, sei es Diskette, Cassette oder Drucker, ausgegeben werden, so legt der Computer im Video-RAM einen Pufferspeicher an. In diesen werden solange Daten eingeschrieben, bis er voll ist. Erst dann werden die Informationen zum jeweiligen Peripheriegerät geschickt.

Beim Cassettenrecorder sieht das dann so aus: Sobald ein Datensatz übermittelt wird, geht er zunächst in den Pufferspeicher und sofort danach an den Cassettenrecorder. Bevor er jedoch gespeichert wird, wird ein Pfeifton aufgezeichnet, der zur Synchronisation beim erneuten Einlesen der Daten dient. Da nun jeder einzelne Datensatz, sei er ein Byte oder 255 Bytes lang, so übermittelt wird, addiert sich das bei größeren Datenfeldern zu unerträglichen Wartezeiten von bis zu 30 Minuten. Nicht zuletzt gibt es auch keine Möglichkeit, die übertragenen Daten zu prüfen (VERIFY).

Speichert man hingegen ein Basic-Programm auf Cassette ab, so geht das in erstaunlich kurzer Zeit und die Aufzeichnung läßt sich sofort danach auf Schreibfehler prüfen. Auch ist es auffällig, daß nur am Anfang der Aufzeichnung der Pfeifton gesendet wird – im Gegensatz zur Datenspeicherung, bei der dies ja vor jedem einzelnen Satz geschieht.

Die Lösung ist ganz einfach: Der Pufferbereich im Video-RAM ist bei der Programmaufzeichnung genauso lang wie das Programm – also maximal ca. 13 KByte, und nicht 255 Bytes wie sonst. Warum kann das Betriebssystem dann Daten nicht ebenfalls in einem Riesenpuffer unterbringen und dann auf einen Schlag speichern? Nun, in den 16 KByte des Video-RAMs belegen die Bildschirminformationen nur rund 3 KByte, der Rest wird vom Computer zur Stringspeicherung verwendet. Daher kommt es auch, daß nach dem SAVEn eines Programms plötzlich alle Variablen verschwunden sind. Sie werden sicherheitshalber vom Betriebssystem gelöscht, da nicht gewährleistet werden kann, daß sie nicht vom Programm-Puffer überschrieben wurden. Bei der Datenspeicherung hingegen müssen selbstverständlich alle Variablen erhalten werden, so daß irgendwo im Video-RAM 255 Bytes gesucht und die Daten von dort aus zur Cassette geschickt werden.

Das Programm SUPERFILE macht sich diese Besonderheiten des TI-Betriebssystems zunutze und speichert einen gesamten String-Array en bloc auf Cassette. Dazu ist es allerdings nötig, den Speicher mehrfach umzukopieren. Das dauert selbst in Maschinensprache noch einige Sekunden. Noch eine Vorbemerkung: Im High Memory müssen noch mindestens 8 KByte Speicherplatz frei sein. Dies können Sie mit dem SIZE-Befehl feststellen: Der Wert vor „BYTES OF PROGRAM SPACE FREE“ muß größer als 8000 sein. Außerdem kann der Low Memory nicht für weitere Maschinenprogramme benutzt werden.

Tippen Sie den Basic-Loader ab und speichern Sie diesen auf Kassette. Um damit arbeiten zu können, muß der Basic-Loader anschließend geladen und gestartet werden. Steht das Maschinenprogramm im Speicher, dann haben Sie zwei neue Befehle, mit denen Sie einen gesamten String-Array auf Cassette speichern und von dort wieder laden können.

Der erste Befehl heißt:

```
CALL LINK("CSAVE",Array$
      (),Count)
```

Für Array$() setzen Sie hier den Namen des gewünschten Datenfelds ein, für Count die Anzahl der zu speichernden Elemente des Stringarrays. Für Count = 150 werden dann die Feldelemente 0 bis 150 abgespeichert. Beachten Sie hier auch, daß nach dem Stringnamen leere Klammern folgen. Keinesfalls dürfen Sie in die Klammern eine Zahl oder Variable einsetzen, da dann unweigerlich eine Fehlermeldung ausgegeben wird.

Eingelesen werden die Daten mit:

```
CALL LINK("CLOAD",
      Array$())
```

Hier erkennt der Computer automatisch die Zahl der Datensätze. Die Namen der Datenfelder beim Speichern und Laden dürfen natürlich unterschiedlich sein. Haben Sie vergessen, vor dem Laden das Datenfeld mit DIM richtig zu dimensionieren, erscheint hier natürlich eine Fehlermeldung.

Alles klar? Hier noch zwei kurze Beispielprogramme:

```
100 ! SPEICHERN
110 DIM A$(200)
120 FOR I=1 TO 200
130 A$(I)="DIES IST DER
    DATENSATZ "&STR$(I)
140 NEXT I
150 CALL LINK("CSAVE",A$
    (),200)
160 END
```

Lassen Sie das Programm ablaufen, dann erscheinen nach kurzer Zeit die Meldungen zur Bedienung des Cassettenrecorders – sie funktionieren genauso wie beim Speichern eines Basic-Programms. Auch hier können Sie nach dem Speichern bei der Meldung "Check Tape (Y or N)?" entscheiden, ob die Daten geprüft werden sollen. Sie brauche wieder nur die Anweisungen des Computers zu befolgen. Sobald wieder READY erscheint, löschen Sie das Programm mit NEW und geben Sie das folgende Programm ein:

```
100 ! LADEN
110 DIM A$(200)
120 CALL LINK("CLOAD",
    A$())
130 FOR I=1 TO 200
140 PRINT A$(I)
150 NEXT I
160 END
```

Starten Sie das Programm und spulen Sie die Cassette an den Beginn der Aufzeichnung zurück. Nach kurzer Zeit erscheinen wieder die vorher gespeicherten Datensätze. Sie sehen, das Maschinenprogramm ist sehr leicht zu bedienen und die neuen Befehle sind um vieles komfortabler als die Basic-Pendants. Dort müßten Sie erst die Datei im richtigen Format eröffnen, mit einer FOR-NEXT Schleife die Daten schreiben, am Ende einen End-of-File Text improvisieren und die Datei wieder schließen. Ähnlich umständlich ist das erneute Einlesen der Daten.

Nun für Maschinensprache Freaks Erläuterungen zum Programm:

Das Problem bei der Programm-Entwicklung war, daß der Video-RAM zwar schon mit den benötigten Strings belegt ist, aber auch mit einer Meng

anderer Daten: weiteren Strings, DEF-Definitionen, Unterprogrammlisten, Variablenlisten, Datei-Puffern usw. Man kann also nicht einfach den ganzen Video-RAM speichern und später wieder laden; das ladende Programm würde mit Sicherheit abstürzen, wenn plötzlich völlig andere Variablenlisten und Definitionen im Speicher stünden.
So sind mehrere Umkopiervorgänge in sämtlichen RAM-Bereichen notwendig:

SPEICHERN:
- 6 KBytes des High Memory in den freien Low Memory kopieren, die übrigen 8 KByte, die frei bleiben müssen, können hier nicht mehr gesichert werden.
- Die Variablen werden mit STRREF (String Reference) aus Basic geholt und in die insgesamt 14 (6+) KBytes des High Memory geschrieben.
- Jetzt wird der Inhalt des High Memory und des Video-RAMs byteweise ausgetauscht und die Daten vom Video-RAM aus auf Cassette geschrieben.
- Zum Schluß wird aus den 14 KByte des High Memory der frühere Inhalt des Video-RAMs wieder in den Video-RAM geschickt. Außerdem werden die im Low Memory gespeicherten 6 KBytes des High Memory dorthin zurückkopiert.

LADEN:
- Zuerst werden wieder 6 KByte des High Memory in den Low Memory kopiert, auch hier können die übrigen benötigten 8 KByte nicht gesichert werden.
- In einem Block wird der gesamte Video-RAM in den High Memory kopiert, worauf die Daten als Block von Cassette in den Video-RAM geholt werden.
- Nun wird Byte für Byte der Inhalt des mit den Informationen des Video-RAMs belegten High Memory und der Video-RAM ausgetauscht. Das heißt, daß jetzt der alte Inhalt des Video-RAMs wieder da ist und sämtliche eingelesenen Daten im High Memory stehen.

- Mit dem Hilfsprogramm STRASG (String Assignment) wird nun String für String wieder an den Basic-Interpreter übergeben. Zu guter Letzt wird der 6K-Bereich aus dem Low Memory wieder in den High Memory zurückgelegt.

Von diesen - zugegeben komplizierten - Vorgängen erfährt der Benutzer natürlich nichts, die beiden Unterprogramme CSAVE und CLOAD erledigen diese mühesame Arbeit automatisch für ihn - und zwar mit Hochgeschwindigkeit; die Beispielprogramm werden in rund 1 Minute 15 Sekunden bewältigt - gegenüber rund 20 Minuten im normalen Aufzeichnungsverfahren!

```
100 !  ********************
110 ! **********************
120 ! *                    *
130 ! *     SUPERFILE      *
140 ! *                    *
150 ! **********************
160 !  ********************
170 !
180 ! ----------------------
190 ! (C) Martin Kotulla 85
200 ! ----------------------
210 !
220 CALL CLEAR :: CALL HCHAR(1,5,42,25)
230 CALL CHAR(45,"00000000FF")
240 CALL CHAR(128,"DFBF60C0919B959191808080C060BFDF")
250 CALL CHAR(130,"FBFD060301010149516151490306FDFB")
260 CALL HCHAR(2,4,42,27):: CALL HCHAR(3,4,42):: CALL HCHAR(3,30,42)
270 DISPLAY AT(4,2):"*          SUPERFILE      *"
280 CALL MAGNIFY(3):: CALL SPRITE(#1,128,2,22,80)
290 CALL HCHAR(5,4,42):: CALL HCHAR(5,30,42):: CALL HCHAR(6,4,42,27)
300 CALL HCHAR(7,5,42,25)
310 CALL CHAR(91,"7E8199A1A199817E")
320 DISPLAY AT(10,5):"[ MARTIN KOTULLA 1985" :: CALL HCHAR(12,7,45,21)
330 DISPLAY AT(15,1):"LADEN:"
340 DISPLAY AT(17,1):"CALL LINK(""CLOAD"",A$())"
350 DISPLAY AT(20,1):"SPEICHERN:"
360 DISPLAY AT(22,1):"CALL LINK(""CSAVE"",A$(),NUM)"
370 CALL INIT
380 FOR I=9460 TO 10034 STEP 4 :: READ A,B,C,D :: CALL LOAD(I,A,B,C,D)
390 NEXT I :: CALL LOAD(8194,63,255,63,240)
400 CALL LOAD(16368,67,76,79,65,68,32,37,50,67,83,65,86,69,32,37,154)
410 CALL CLEAR :: END
420 !
430 DATA 255,000,000,000,000,003,004,000,008,000,000,058,000,255,006,012
440 DATA 009,196,080,080,016,000,096,003,067,083,049,000,255,255,205,096
450 DATA 033,131,035,253,038,184,040,183,041,182,042,195,043,193,044,179
```

```
460 DATA 045,194,047,196,058,181,059,180,060,191,061,190,062,192,002,224
470 DATA 037,018,002,006,054,059,200,006,037,008,006,160,038,062,002,000
480 DATA 009,196,002,001,160,000,002,002,054,059,004,032,032,044,002,000
490 DATA 003,208,002,001,005,000,004,032,032,032,216,032,036,246,131,124
500 DATA 004,032,038,186,000,061,006,160,038,152,004,192,002,001,000,001
510 DATA 002,002,160,000,209,018,009,132,004,032,032,016,160,132,005,130
520 DATA 136,018,037,000,019,006,136,034,000,002,037,000,019,002,005,128
530 DATA 016,241,004,096,038,030,002,224,037,018,004,192,002,001,000,002
540 DATA 004,032,032,012,004,032,032,024,018,184,193,160,131,074,006,160
550 DATA 038,062,004,192,002,001,000,001,002,002,160,000,004,199,212,160
560 DATA 036,244,004,032,032,020,208,210,009,131,161,195,005,135,160,131
570 DATA 005,130,005,128,129,128,018,243,002,039,000,004,005,130,196,160
580 DATA 037,000,002,004,003,214,192,135,192,071,006,160,035,230,006,160
590 DATA 038,152,002,000,003,208,002,001,006,000,004,032,032,032,216,032
600 DATA 036,246,131,124,004,032,038,186,000,061,002,000,009,196,002,001
610 DATA 160,000,002,002,054,059,004,032,032,036,002,000,039,116,002,001
620 DATA 190,002,002,002,012,028,204,112,006,002,024,253,004,032,038,186
630 DATA 000,052,002,224,131,224,004,096,000,112,002,000,003,208,002,001
640 DATA 037,002,002,002,000,018,004,032,032,036,002,008,191,064,136,008
650 DATA 131,134,026,004,002,000,011,000,004,032,032,052,200,032,037,012
660 DATA 131,074,216,032,037,014,131,076,002,000,003,221,200,000,131,086
670 DATA 200,032,036,248,131,084,004,224,131,208,216,032,036,252,131,109
680 DATA 002,000,039,116,002,001,190,002,002,002,012,028,204,049,006,002
690 DATA 024,253,004,091,002,000,009,196,002,003,160,000,002,005,054,059
700 DATA 004,032,032,040,193,001,208,083,004,032,032,032,220,196,005,128
710 DATA 006,005,022,246,004,091,037,018,038,190,216,032,152,002,063,222
720 DATA 216,032,152,002,063,223,006,032,063,222,216,062,156,002,216,062
730 DATA 156,002,002,224,131,224,002,000,131,230,002,001,063,182,002,002
740 DATA 000,013,204,112,006,002,022,253,184,032,036,250,131,115,208,160
750 DATA 131,115,009,130,002,034,130,254,002,000,039,012,006,064,204,128
760 DATA 196,160,036,254,004,096,000,112,006,096,131,115,216,032,063,222
770 DATA 156,002,216,032,063,223,156,002,002,000,131,230,002,001,063,182
780 DATA 002,002,000,013,204,049,006,002,022,253,002,224,037,018,003,128
790 !
800 ! PRUEFSUMME:
810 !
820 RESTORE
830 FOR I=0 TO 572 STEP 4 :: READ A,B,C,D :: SUM=SUM+A+B+C+D
840 NEXT I :: DISPLAY BEEP:"DIE DATA-ZEILEN SIND"
850 IF SUM=41849 THEN PRINT "KORREKT!" ELSE PRINT "FEHLERHAFT!"
860 END
```

```
4818- 30 3D 4C 41 4E 47 53 41
4820- 4D 29 3A 22 3B 47 00 3A
4828- 0E D6 0B AD 47 D1 31 CE
4830- 47 CF 31 30 C4 33 30 32
4838- 30 00 45 0E E0 0B B9 32
4840- 35 35 2C 47 00 4D 0E EA
4848- 0B AB 31 35 00 53 0E A0
4850- 0F B2 00 59 0E B4 0F 97
4858- 00 7C 0E BE 0F 84 22 41
```

```
4860- 42 53 50 45 49 43 48 45
4868- 52 4E 20 2D 20 46 49 4C
4870- 45 4E 41 4D 45 3A 22 3B
4878- 46 49 24 00 8D 0E C8 0F
4880- AD 46 49 24 D0 22 22 C4
4888- 34 30 32 30 00 B4 0E D2
4890- 0F BA E7 28 34 29 3B 22
4898- 42 53 41 56 45 22 3B 46
```

Eingabemodul für den TI-99/4A mit oder ohne X-Basic

Diese Schaltung ermöglicht das Betreiben eines I/O-Ports auch ohne 32K-Erweiterung und ohne X-Basic

Eine solche 8-Bit-Eingabe ist nur über den Joystickport möglich, und zwar in der Form 2x4 Bit. Es können auch 1x6 Bit oder andere Kombinationen gewählt werden, je nach Einsatzgebiet.

Die beiden Eingänge für die Feuerknöpfe legen die beiden 4-Bit-Kombinationen fest, die angesprochen werden sollen. Dabei muß man aber sicherstellen, daß nur ein Eingang geschaltet ist, da sich sonst die Feuerknopfeingänge gegenseitig sperren und somit auch kein X/Y-Signal durchdingt.

Zum Schaltplan:

Die beiden Datenbits D0 und D1 werden also zur Ansteuerung der beiden Bitblöcke $\overline{D2}$-$\overline{D5}$ und D2-D5 benutzt.

Wenn D0 angesteuert ist, dann liest der Rechner den Datenblock D2-D5. Wenn D1 angesteuert ist, dann liest der den Datenblock $\overline{D2}$-$\overline{D5}$.

Zur linken Seite der Schaltung ist nur soviel zu sagen, daß dort die Kontakte sitzen, die die Kanäle schalten. Wenn man Eingänge mit niedrigen Eingangsspannungen hat, muß über eine kleine Verstärkerschaltung das entsprechende Relais angesteuert werden.

Wenn man aber z. B. wissen will, wo im Haus das Licht brennt, muß man direkt an den zu prüfenden Verbraucher ein 220-V-Relais klemmen und die Schaltkontakte, wie auf der linken Seite der Schaltung, anklemmen. Wem Relais zu laut sind, der kann natürlich auch Optokoppler als Schaltelemete nehmen oder einen Transistor benutzen.

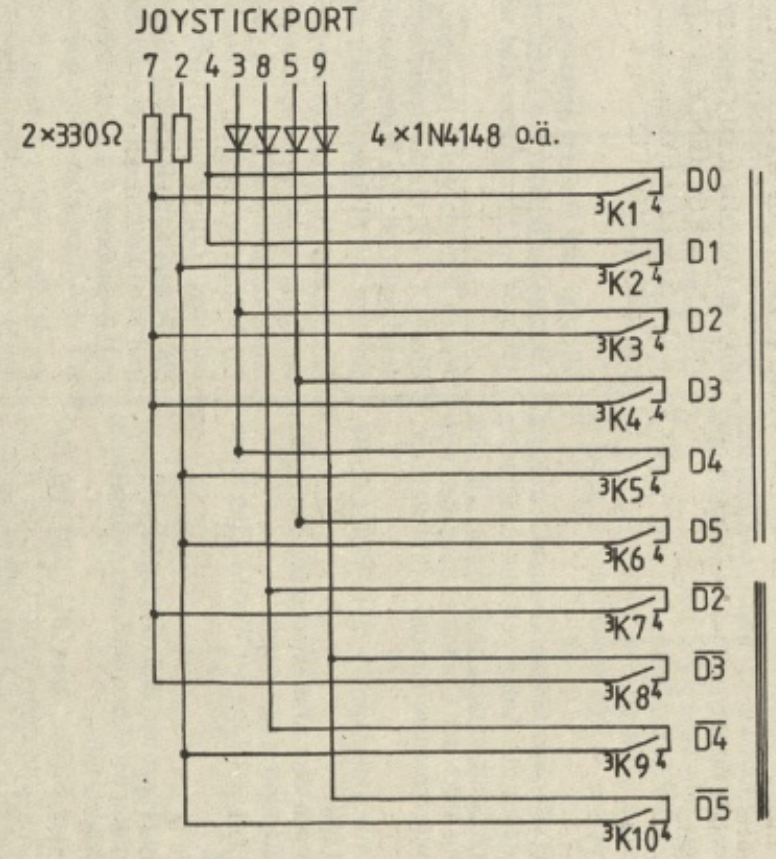
Die gesamten Teile, die man für die Schaltung benötigt, sind bereits eingebaut, außer der 9-poligen Trapezbuchse für den Joystickport.

Die Software zur Ansteuerung:

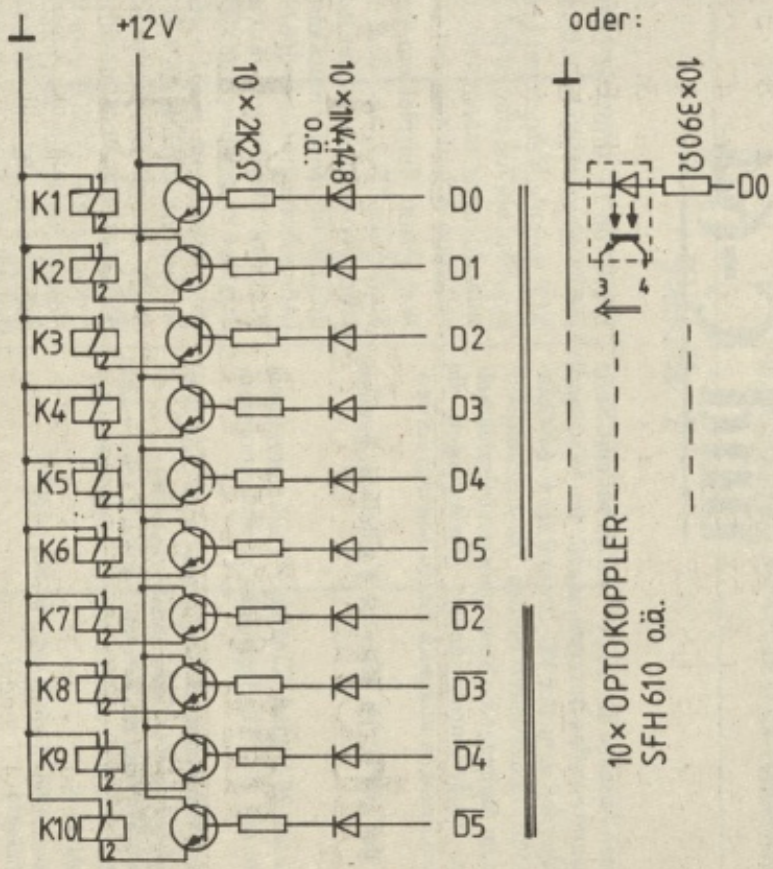
```
100 REM ABFRAGE OB DO ODER D1 GESETZT SIND MIT
    SPRUNG IN DIE ENTSPRECHENDE ROUTINE
110 CALL KEY(1,K1,S1):: CALL KEY(2,K2,S2)::
    IF K1=18 THEN GOSUB 1000 ELSE IF K2 =18
    GOSUB 2000
120 GOTO 110
999 REM ROUTINE ZUM EINLESEN DER WERTE D2-D5
1000 CALL JOYST(1.X1,Y1):: CALL JOYST(2,X2,Y2)::
    IF X1=4 THEN D2=1 ELSE D2=0
1010 IF X1=4 THEN D3=1 ELSE D3=0
1020 IF Y2=4 THEN D4=1 ELSE D4=0
1030 IF X2=4 THEN D5=1 ELSE D5=0
1040 REM DIE DATEN LIEGEN JETZT IN D2-D5 ZUR
    VERFÜGUNG
1050 RETURN
1999 REM ROUTINE ZUM EINLESEN DER WERTE D2-D5
2000 CALL JOYST(1,X1,Y1):: CALL JOYST(2,X2,Y2)::
    IF Y1=-4 THEN DD2=1 ELSE DD2=0
2010 IF X1=-4 THEN DD3=1 ELSE DD3=0
2020 IF X2=-4 THEN DD4=1 ELSE DD4=0
2030 IF X2=-4 THEN DD5=1 ELSE DD5=0
2040 REM DIE DATEN LIEGEN JETZT IN DD2-DD5 ZUR
    VERFÜGUNG
2050 RETURN
```

Das Programm läßt sich leicht auf TI-Basic umschreiben und natürlich auch erweitern. Der Phantasie sind hier keine Grenzen gesetzt. Wem diese Eingabemethode dennoch zu kompliziert ist, der nimmt einfach nur die Kanäle D1-D5 und hat so reale 5 Bit paralell.

Eingabemodul für den TI-99/4A
mit oder ohne X-Basis V 1.0



7 JOYST 1
2 JOYST 2
4 FIRE
3 UP
8 DOWN
5 LEFT
9 RIGHT



10 × BC 107 o.ä.

Tips und Tricks für den TI-99/4 A Extended Basic

1. Listschutz

```
100 ON BREAK NEXT
110 CALL CLEAR ::
    INPUT CODE$
120 IF CODE$="TEST"
    THEN 1000
130 ON ERROR 150
140 RUN ""
150 RUN
1000 REM Hier beginnt das
    eigentl. Programm
```

Das Programm wird mit SAVE "CS1", PROTECTED abgespeichert. Es kann nun nicht mehr gelistet werden. Durch den Befehl in Zeile 100 kann das Programm auch nicht mehr gestoppt werden.

In Zeile 110 wird der Benutzer aufgefordert, den Code einzugeben. Wird das richtige Codewort eingegeben (hier "TEST"), dann verzweigt das Programm zum eigentlichen Programmbeginn in Zeile 1000.
Wird aber das falsche Codewort eingegeben, so ist das Programm verloren: Die Fehlermeldung "NO PROGRAMM PRESENT" wird ausgegeben.

2. Echtes Charset

```
1000 SUB NEUDEF
1010 FOR Z=65 TO :: CALL
     CHARPAT(Z,A$)
1020 CHAR$="0000"&SEG$
     (A$,1,4)&SEG$(A$,7,4)
     &SEG$(A$,13,4)
1030 CALL CHAR(Z+32,
     CHAR$) :: NEXT Z
1040 CALL CHAR(99,"000000
     3C4040403C",106,"000000
     080808483000000024283
     02824")
1050 SUBEND
```

Der Befehl "CALL CHARSET" soll beim TI eigentlich alle Zeichen in Ihre ursprüngliche Form zurückdefinieren, falls sie durch "CALL CHAR" umdefiniert wurden. Dies funktioniert aber nur bei den Großbuchstaben. Die Kleinbuchstaben kann man nun durch "CALL NEUDEF" ebenfalls in ihrer ursprünglichen Form definieren.

Nützliche BASIC-Erweiterung für den TI-99

Das Programm STRINGKIT erweitert das Extended Basic des TI-99 um einige nützliche Stringfunktionen, die der Programmierer bisher schmerzlich vermißte. Das Programm ist in Maschinensprache geschrieben und damit auf jeden Fall schneller als Basic-Unterprogramme, die diese Befehle simulieren. STRINGKIT läuft auf dem TI-99/4A mit Extended Basic und 32K-Speichererweiterung.

Bei der Programmeingabe gibt es wie üblich drei Wege: Eingabe des Quellcodes und Assemblieren mit dem Editor/Assembler-Modul, Speichern des Objektcodes in einer Datei des Formats DISPLAY/FIXED 80 oder Eingabe des Basic-Loaders mit den DATA-Zeilen. Allen Besitzern des Editor/Assemblers sei einer der ersten beiden Wege empfohlen, da das Programm als Objektcode im Speicher frei verschiebbar (relokatibel) ist, während die Version mit DATA-Zeilen an den Speicherbereich 9460 bis 10495 gebunden ist.

Entsprechend der Programmspeicherung erfolgt das Einlesen mit CALL LOAD("Datei") oder RUN "Programm". Steht das Programm im Speicher, so wird das Basic um sechs Befehle erweitert, die alle mit CALL LINK aufgerufen werden:

CALL LINK("UPPER$",IN$,OUT$) wandelt im String IN$ alle Kleinbuchstaben in Großbuchstaben um und speichert den neuen String in OUT$. Alle anderen Zeichen im String bleiben unverändert. Der Befehl entspricht der UPPER$-Funktion im CPC 464-Basic und ist nützlich bei Benutzereingaben, die in ein bestimmtes Format gebracht werden sollen.

Beispiel: CALL LINK("UPPER$","ABCabc12",B$):: PRINT B$ liefert "ABCABC12"

CALL LINK("LOWER$",IN$,OUT$) wandelt analog zum UPPER$-Befehl alle Großbuchstaben des Strings IN$ in Kleinbuchstaben um und speichert das Ergebnis in OUT$. Dieser Befehl entspricht der LOWER$-Funktion im CPC 464-Basic.

Beispiel: CALL LINK("LOWER$","ABCabc1;",B$):: PRINT B$ liefert "abcabc12"

CALL LINK("LEFT$",IN$,COUNT,OUT$) entspricht der LEFT$-Funktion im Microsoft-Basic und vielen anderen Basic-Dialekten. Der Befehl nimmt aus dem String IN$ alle Zeichen vom ersten bis zum durch COUNT angegebenen Zeichen und speichert sie in OUT$.

Beispiel: CALL LINK("LEFT$","ABCDEFG",3,B$):: PRINT B$ liefert "ABC"

CALL LINK("RIGHT$",IN$,COUNT,OUT$) entspricht der RIGHT$-Funktion im Microsoft-Basic und nimmt im Gegensatz zu LEFT$ die angegebene Anzahl von Zeichen rechts vom String IN$ ab.

Beispiel: CALL LINK("RIGHT$",ABCDEFG",4,B$):: PRINT B$ liefert "DEFG"

CALL LINK("SWAP",A$(),B$(), COUNT) tauscht die Elemente zweier Stringdatenfelder gegeneinander aus. Dabei werden alle Elemente vom 0. bis 1. Element (entsprechend OPTION BASE) bis zu dem Wert von COUNT entsprechenden Element berücksichtigt. Das Programm beachtet also, ob OPTION BASE 0 oder OPTION BASE 1 gewählt wurde. Bitte beachten Sie, daß den Array-Bezeichnungen leere Klammern folgen müssen, Zahlenwerte oder Variablennamen in den Klammern führen zu einer Fehlermeldung.

CALL LINK("SWAP",A(),B(), COUNT) ist das zweite Format des SWAP-Befehls, denn hiermit können auch die Inhalte von Zahlenfeldern ausgetauscht werden. Auch hier gilt, daß der für OPTION BASE gewählte Wert berücksichtigt wird.

Für Maschinenprache-Programmierer sei noch erwähnt, wie STRINGKIT auf den Wert von OPTION BASE kommt. Texas Instruments hat es nirgendwo erwähnt, aber per Zufall entdeckte der Autor in Disassemblieren die Speicherstelle >8343. Sie enthält bei OPTION BASE 0 den Wert 0, bei OPTION BASE 1 entsprechend die Zahl 1. Jeder andere Wert der hier gePOKEt wird, entspricht ebenfalls OPTION BASE 1.

Martin Kotulla

```
100 !  ******************
110 ! ********************
120 ! *                  *
130 ! *    STRINGKIT     *
140 ! *                  *
150 ! ********************
160 !  ******************
170 !
180 ! ----------------------
190 ! (C) Martin Kotulla 85
200 ! ----------------------
210 !
```

```
220 CALL CHAR(42,"AA55AA55AA55AA55",91,"7E8199A1A199817E")
230 CALL CHAR(45,RPT$("0",8)&"FF")
240 CALL CHAR(128,"DFBF60C0919B959191808080C060BFDF")
250 CALL CHAR(130,"FBFD0603010101495161514903O6FDFB")
260 CALL CLEAR :: CALL HCHAR(2,4,42,26)
270 CALL VCHAR(3,4,42,5):: CALL VCHAR(3,29,42,5)
280 CALL MAGNIFY(3):: CALL SPRITE(#1,128,2,28,70)
290 DISPLAY AT(5,12)SIZE(9):"STRINGKIT"
300 CALL HCHAR(8,4,42,26)
310 DISPLAY AT(12,4):CHR$(91)&"  MARTIN KOTULLA 1985"
320 CALL HCHAR(13,6,45,22):: DISPLAY AT(17,7):"DATAs PRUEFEN? J"
330 ACCEPT AT(17,22)BEEP SIZE(-1)VALIDATE("JN"):JN$
340 IF JN$="N" THEN 390
350 FOR I=1 TO 1080 STEP 2 :: READ A1,A2 :: B=B+A1+A2 :: NEXT I
360 DISPLAY AT(20,5):"DIE DATA-ZEILEN SIND"
370 IF B<>90212 THEN DISPLAY AT(22,10):"FEHLERHAFT!" :: END
380 DISPLAY AT(22,11):"KORREKT !"
390 CALL PEEK(8198,A):: IF A<>170 THEN CALL INIT :: RESTORE ELSE RESTORE
400 FOR I=9460 TO 10495 STEP 5 :: READ A,B,C,D,E
410 CALL LOAD(I,A,B,C,D,E):: NEXT I
420 CALL PEEK(8196,A,B):: C=A*256+B-40
430 CALL LOAD(8196,INT(C/256),C AND 255)
440 FOR I=C TO C+39 :: READ A :: CALL LOAD(I,A):: NEXT I
450 DISPLAY ERASE ALL BEEP :: END
460 !
470 ! MASCHINENCODE:
480 !
490 DATA 255,000,000,000,000,000,000,000,000,000,000,000,203,020,203,053
500 DATA 203,078,203,231,204,071,204,150,204,228,205,029,205,075,205,096
510 DATA 033,131,035,253,038,184,040,183,041,182,042,195,043,193,044,179
520 DATA 045,194,047,196,058,181,059,180,060,191,061,190,062,192,094,197
530 DATA 255,058,058,130,065,084,240,071,079,133,073,070,132,079,078,155
540 DATA 079,082,186,080,073,221,084,079,177,255,065,066,083,203,065,076
550 DATA 076,236,065,078,068,187,065,083,067,220,065,084,078,204,066,089
560 DATA 069,003,067,079,078,001,067,079,083,205,068,069,070,137,068,073
570 DATA 077,138,069,078,068,139,069,079,070,202,069,088,080,206,070,079
580 DATA 082,140,073,078,084,207,076,069,078,213,076,069,084,141,076,079
590 DATA 071,208,077,065,088,223,077,073,078,224,078,069,087,000,078,079
600 DATA 084,189,078,085,077,004,079,076,068,005,080,079,083,217,082,069
610 DATA 067,222,082,069,077,154,082,069,083,006,082,078,068,215,082,085
620 DATA 078,169,083,071,078,209,083,073,078,210,083,081,082,211,083,085
630 DATA 066,161,084,065,066,252,084,065,078,212,086,065,076,218,088,079
640 DATA 082,188,255,066,065,083,069,241,066,069,069,080,238,067,065,076
650 DATA 000,255,067,072,082,036,214,068,065,084,065,147,038,066,200,001
660 DATA 037,178,200,002,037,180,004,096,048,094,004,224,038,066,004,096
670 DATA 048,124,004,192,002,008,224,129,216,008,131,212,002,008,160,129
680 DATA 216,008,140,002,006,200,216,008,140,002,004,224,038,066,004,224
690 DATA 034,160,192,000,019,002,004,096,034,018,004,096,034,032,000,000
700 DATA 007,096,047,164,022,007,007,032,034,152,200,001,038,086,004,032
710 DATA 043,074,000,000,004,091,007,032,034,156,006,160,036,058,006,160
720 DATA 038,106,004,096,035,016,007,096,047,164,022,244,007,032,034,152
730 DATA 002,009,041,172,002,008,032,032,002,006,000,080,222,072,006,006
740 DATA 021,253,200,001,038,142,004,032,043,116,000,000,041,172,192,000
750 DATA 019,225,004,096,037,184,200,004,034,158,006,160,036,052,193,032
760 DATA 034,158,004,224,034,158,160,135,016,220,007,032,034,156,006,160
```

```
770 DATA 036,058,006,001,128,193,017,006,002,034,255,170,004,224,034,152
780 DATA 004,096,035,016,192,195,022,001,004,193,192,193,016,249,000,000
790 DATA 000,007,000,012,000,025,000,030,000,045,000,059,000,079,255,255
800 DATA 255,255,255,255,255,255,255,255,255,255,255,255,255,255,255,255
810 DATA 255,255,255,000,000,006,002,011,035,016,194,002,098,032,034,142
820 DATA 194,072,098,004,017,006,200,008,038,248,002,136,000,040,020,001
830 DATA 004,091,004,196,002,224,038,248,006,160,040,152,002,001,000,002
840 DATA 006,160,040,138,192,096,131,074,209,032,036,244,009,132,129,001
850 DATA 018,002,004,096,040,198,006,193,216,001,036,244,002,002,036,244
860 DATA 004,096,040,212,002,224,038,248,006,160,040,152,002,001,000,002
870 DATA 006,160,040,138,192,096,131,074,208,160,036,244,009,130,128,066
880 DATA 020,002,004,096,040,198,096,129,002,004,036,244,160,132,006,193
890 DATA 212,129,004,096,040,212,002,224,038,248,006,160,040,152,002,000
900 DATA 036,245,208,096,036,244,009,129,208,144,009,130,002,130,000,097
910 DATA 017,005,002,130,000,122,021,002,002,034,255,224,006,194,220,002
920 DATA 006,001,024,242,004,192,002,001,000,002,002,002,036,244,004,032
930 DATA 032,016,004,096,040,222,002,224,038,248,006,160,040,152,002,000
940 DATA 036,245,208,096,036,244,009,129,208,144,009,130,002,130,000,065
950 DATA 017,005,002,130,000,090,021,002,002,034,000,032,006,194,220,002
960 DATA 006,001,024,242,004,192,002,001,000,002,002,002,036,244,004,032
970 DATA 032,016,004,096,040,222,002,224,038,248,002,001,000,003,006,160
980 DATA 040,138,193,160,131,074,210,032,131,000,009,136,002,136,000,005
990 DATA 019,025,208,032,131,067,009,128,006,065,004,032,032,012,006,160
1000 DATA 040,168,005,129,004,032,032,012,006,001,004,032,032,008,006,160
1010 DATA 040,178,005,129,004,032,032,008,005,128,006,001,006,006,024,237
1020 DATA 004,096,040,222,192,198,208,032,131,067,009,128,002,001,000,001
1030 DATA 002,002,036,244,212,160,038,246,004,032,032,020,005,129,002,002
1040 DATA 037,245,212,160,038,246,004,032,032,020,006,001,004,032,032,016
1050 DATA 005,129,002,002,036,244,004,032,032,016,005,128,006,001,006,003
1060 DATA 024,231,004,096,040,222,004,192,004,032,032,012,004,032,032,024
1070 DATA 018,184,004,091,004,192,002,001,000,001,002,002,036,244,004,032
1080 DATA 032,020,004,091,002,003,131,074,002,004,036,245,016,004,002,003
1090 DATA 036,245,002,004,131,074,002,005,000,004,205,051,006,005,022,253
1100 DATA 004,091,216,032,038,246,036,244,002,000,030,000,004,032,032,052
1110 DATA 004,192,002,001,000,003,004,032,032,016,216,032,038,246,036,244
1120 DATA 216,032,038,246,037,245,216,032,038,247,131,124,002,224,131,224
1130 DATA 004,096,000,112,040,020,200,012,041,014,016,002,200,008,041,014
1140 !
1150 ! NAMENSTABELLE:
1160 !
1170 DATA 083,087,065,080,032,032,039,250,076,079,087,069,082,036,039,186
1180 DATA 085,080,080,069,082,036,039,122,082,073,071,072,084,036,039,072
1190 DATA 076,069,070,084,036,032,039,024
1200 END ! ***********************************************************
```

```
*  ******************************
* ********************************
* *                              *
* *          STRINGKIT           *
* *                              *
* ********************************
*  ******************************
*
*
*  (C)  Martin Kotulla - 7.5.1985
*  ------------------------------
```

```
* Neue Befehle:
*
* - CALL LINK("UPPER$",in$,out$)
* - CALL LINK("LOWER$",in$,out$)
*
* - CALL LINK("LEFT$",in$,cnt,out$)
* - CALL LINK("RIGHT$",in$,cnt,out$)
*
* - CALL LINK("SWAP",io1(),io2(),cnt)
* - CALL LINK("SWAP",io1$(),io1$(),cnt)
*
* in$          = Eingabestring
* out$         = Ausgabestring
* cnt          = Zähler
* io1/io2      = Eingabe und Ausgabe
* io1$/io2$    = Ein- und Ausgabestring
*

       DEF  LEFT$,RIGHT$
       DEF  UPPER$,LOWER$
       DEF  SWAP

       IDT  'STRKIT'

NEXT   EQU  >0070

ARGIDT EQU  >8300          Argument-Identifier bei CALL LINK-Aufruf
OPBASE EQU  >8343          Wert von Option Base
FAC    EQU  >834A          Fließkomma-Akkumulator
STATUS EQU  >837C          GPL-Statusbyte
GPLWS  EQU  >83E0          GPL-Workspace

NUMASG EQU  >2008          Numerischen Parameter an Basic übergeben
NUMREF EQU  >200C          Numerischen Parameter aus Basic holen
STRASG EQU  >2010          String an Basic übergeben
STRREF EQU  >2014          String aus Basic holen
XMLLNK EQU  >2018          ROM-Routinen aufrufen
ERR    EQU  >2034          Fehlermeldung ausgeben

ERRBV  EQU  >1E00          Code für BAD VALUE
CFI    EQU  >12B8          Convert Floating Point To Integer

BUF1   BYTE 255
BUF2   BSS  256
BUF3   BYTE 255
BUF4   BSS  256

HEXFF  BYTE >FF
HEX00  BYTE >00

USRWS  BSS  32             Workspace des Programms

*******************************************************************************

LEFT$  LWPI USRWS          Eigene Workspace-Register laden
       BL   §GETSTR        String aus CALL LINK holen
```

Achtung! - Achtung!

In dieser Ausgabe Super Software-Angebote!

Mehr darüber auf Seite 2!

```
LFTLEN LI   R1,2          Längenangabe aus CALL LINK holen
       BL   §GETNUM
       MOV  §FAC,R1

LFTCHK MOVB §BUF1,R4      Stringlänge des übergebenen Strings holen
       SRL  R4,8          ... und ins LSByte schieben
       C    R1,R4         Stringlänge kleiner als CALL LINK-Angabe?
       JLE  LFTCUT
       B    §BADVAL       Ja - Bad Value ausgeben

LFTCUT SWPB R1            Dem Längenbyte neuen Wert zuweisen
       MOVB R1,§BUF1
       LI   R2,BUF1       Den String an Basic übergeben
       B    §ASGSTR       Rücksprung nach Basic

*****************************************************************

RIGHT$ LWPI USRWS         Eigene Workspace-Register laden
       BL   §GETSTR       String aus CALL LINK holen

RGTLEN LI   R1,2          Längenwert aus CALL LINK holen
       BL   §GETNUM
       MOV  §FAC,R1

RGTCHK MOVB §BUF1,R2      Tatsächliche Länge des Strings holen
       SRL  R2,8
       C    R2,R1         Mit Wert aus CALL LINK vergleichen
       JHE  RTCALC
       B    §BADVAL       Argument zu klein: Bad Value ausgeben

RTCALC S    R1,R2         Stringlänge:=Stringlänge-Argumentwert
       LI   R4,BUF1
       A    R4,R2         Adresse:=Bufferstart+Stringlänge

RGTCUT SWPB R1            Neue Stringlänge:=Argumentwert
       MOVB R1,*R2        *R2 wird neuer Bufferbeginn
       B    §ASGSTR       String zuweisen, Basic-Rücksprung

*****************************************************************

UPPER$ LWPI USRWS         Eigene Workspace-Register laden
       BL   §GETSTR       String aus CALL LINK holen

UPRLEN LI   R0,BUF2       Zeiger auf den String-Buffer
       MOVB §BUF1,R1      Längenbyte aus dem Buffer holen
       SRL  R1,8          Und ins LSByte schieben

USHTLP MOVB *R0,R2        Einen Buchstaben aus dem Stringbuffer holen
       SRL  R2,8          Und ins LSByte schieben
       CI   R2,'a'        Kleiner als "a"?
       JLT  UNOSHT        Ja - nicht shiften
       CI   R2,'z'        Grösser als "z"?
       JGT  UNOSHT        Ja - nicht shiften

USHIFT AI   R2,-32        ASCII-Code -32 ergibt Großbuchstaben
```

```
UNOSHT SWPB  R2              Buchstaben wieder ins MSByte zurück
       MOVB  R2,*R0+         Und wieder in den Buffer schreiben

USNEXT DEC   R1              Längenzähler dekrementieren
       JOC   USHTLP          Fertig? Nein - weitermachen

UPASG$ CLR   R0              String wieder an Basic zuweisen: einfache Variable
       LI    R1,2            - Zweiter Parameter in CALL LINK
       LI    R2,BUF1         - Zeiger auf Stringbuffer
       BLWP  §STRASG         - String zuweisen
       B     §RETURN         Rücksprung nach Basic

*******************************************************************************

LOWER$ LWPI  USRWS           Eigene Workspace-Register laden
       BL    §GETSTR         String aus CALL LINK holen

LWRLEN LI    R0,BUF2         Zeiger auf den String-Buffer
       MOVB  §BUF1,R1        Längenbyte aus dem Buffer holen
       SRL   R1,8            Und ins LSByte schieben

LSHTLP MOVB  *R0,R2          Einen Buchstaben aus dem Stringbuffer holen
       SRL   R2,8            Und ins LSByte schieben
       CI    R2,'A'          Kleiner als "A"?
       JLT   LNOSHT          Ja - nicht shiften
       CI    R2,'Z'          Grösser als "Z"?
       JGT   LNOSHT          Ja - nicht shiften

LSHIFT AI    R2,32           ASCII-Code +32 ergibt Kleinbuchstaben

LNOSHT SWPB  R2              Buchstaben wieder ins MSByte zurück
       MOVB  R2,*R0+         Und wieder in den Buffer schreiben

LSNEXT DEC   R1              Längenzähler dekrementieren
       JOC   LSHTLP          Fertig? Nein - weitermachen

LWASG$ CLR   R0              String wieder an Basic zuweisen: einfache Variable
       LI    R1,2            - Zweiter Parameter in CALL LINK
       LI    R2,BUF1         - Zeiger auf Stringbuffer
       BLWP  §STRASG         - String zuweisen
       B     §RETURN         Rücksprung nach Basic

*******************************************************************************

SWAP   LWPI  USRWS           Eigene Workspace-Register laden
       LI    R1,3            Zahl der Array-Elemente holen
       BL    §GETNUM
       MOV   §FAC,R6

STRNUM MOVB  §ARGIDT,R8      Argument Identifier holen
       SRL   R8,8            Und ins LSByte schieben
       CI    R8,5            Ist es ein String-Array?
       JEQ   SWAP$           Ja - nach SWAP$ springen

SWPNUM MOVB  §OPBASE,R0      Option Base 0 oder 1
       SRL   R0,8            Ergebnis ins LSByte schieben
       DECT  R1              Zeiger auf erstes Argument in CALL LINK
```

```
SWNULP  BLWP  §NUMREF       Zahl eines Feldelements holen
        BL    §FACBUF       Im internen Buffer zwischenspeichern
        INC   R1            Zeiger auf zweiten Array
        BLWP  §NUMREF       Zahl eines Feldelements aus 2.Array holen
        DEC   R1            Zeiger wieder auf 1.Array
        BLWP  §NUMASG       Und Zahl dem 1.Array zuweisen
        BL    §BUFFAC       Vorherige Zahl aus dem Buffer in den FAC holen
        INC   R1            Zeiger wieder auf 2.Array
        BLWP  §NUMASG       Und Zahl einem Element des 2.Arrays zuweisen

NXTNUM  INC   R0            Die nächsten beiden Elemente der Felder
        DEC   R1            Wieder Zeiger auf erstes Argument
        DEC   R6            Zähler dekrementieren
        JOC   SWNULP        Fertig? Nein - weitermachen
        B     §RETURN       Rücksprung nach Basic

*****************************************************************************

SWAP$   MOV   R6,R3         Zahl der Array-Elemente kopieren
        MOVB  §OPBASE,R0    Option Base 0 oder 1
        SRL   R0,8          Ergebnis ins LSByte schieben
        LI    R1,1          Zeiger auf erstes Argument in CALL LINK

GET1$   LI    R2,BUF1       String aus dem 1.Array in CALL LINK holen
        MOVB  §HEXFF,*R2
        BLWP  §STRREF

GET2$   INC   R1            String aus dem 2.Array in CALL LINK holen
        LI    R2,BUF3
        MOVB  §HEXFF,*R2
        BLWP  §STRREF

GIVE1$  DEC   R1            Zeiger auf 1.Array
        BLWP  §STRASG       Dem 1.Array den String des 2.Arrays zuweisen

GIVE2$  INC   R1            Zeiger auf 2.Array
        LI    R2,BUF1       Zeiger auf String des 1.Arrays
        BLWP  §STRASG       Dem 2.Array den String des 1.Arrays zuweisen

NXTSTR  INC   R0            Zeiger auf nächste Array-Elemente
        DEC   R1            Zeiger auf 1.Array in CALL LINK
        DEC   R3            Schleifenzähler dekrementieren
        JOC   GET1$         Fertig? Nein - weitermachen
        B     §RETURN       Rücksprung nach Basic

*****************************************************************************

GETNUM  CLR   R0            Zahl aus CALL LINK holen
        BLWP  §NUMREF
        BLWP  §XMLLNK       Und in Integer umwandeln
        DATA  CFI
        RT

GETSTR  CLR   R0            String aus CALL LINK holen
        LI    R1,1
        LI    R2,BUF1
```

```
        BLWP  §STRREF
        RT

*****************************************************************************

FACBUF  LI    R3,FAC        Fließkomma-Zahl von FAC in einen Buffer kopieren
        LI    R4,BUF2
        JMP   FTRANS

BUFFAC  LI    R3,BUF2       Fließkomma-Zahl von einem Buffer in FAC kopieren
        LI    R4,FAC
FTRANS  LI    R5,4          8 Bytes = 4*16-Bit Words

FTLOOP  MOV   *R3+,*R4+     Kopierschleife
        DEC   R5            Schleifenzähler herunterzählen
        JNE   FTLOOP        Fertig? Nein - weitermachen
        RT                  Rücksprung zum aufrufenden Programm

*********************************

BADVAL  MOVB  §HEXFF,§BUF1  Maximale Stringlänge in den Buffer schreiben
        LI    R0,ERRBV      Errorcode von BAD VALUE laden
        BLWP  §ERR          Fehlermeldung ausgeben

*********************************

ASGSTR  CLR   R0            String an Basic zuweisen
        LI    R1,3
        BLWP  §STRASG

RETURN  MOVB  §HEXFF,§BUF1    Maximale Stringlänge wiederherstellen
        MOVB  §HEXFF,§BUF3    In beiden Buffern
        MOVB  §HEX00,§STATUS  GPL-Statusbyte löschen
        LWPI  GPLWS           GPL-Workspace laden
        B     §NEXT           Sprung in den GPL-Interpreter

        END

*****************************************************************************
```

Objektcode (Dateiformat DISPLAY/FIXED 80):

```
00404STRKIT  A0000BFF00A0001A0100B00FFA0102A0202BFF00A02047F311F   0001
A0224B02E0C0204B06A0C03A4B0201B0002B06A0C0396BC060B834ABD1207F2DDF 0002
A023AC0000B0984B8101B1202B0460C03D2B06C1BD801C0000B0202C00007F316F 0003
A0250B0460C03E0B02E0C0204B06A0C03A4B0201B0002B06A0C0396BC0607F2F2F 0004
A0266B834ABD0A0C0000B0982B8042B1402B0460C03D2B6081B0204C00007F2FCF 0005
A027CBA084B06C1BD481B0460C03E0B02E0C0204B06A0C03A4B0200C00017F2C9F 0006
A0292BD060C0000B0981BD090B0982B0282B0061B1105B0282B007AB15027F302F 0007
A02A8B0222BFFE0B06C2BDC02B0601B18F2B04C0B0201B0002B0202C00007F2C3F 0008
A02BEB0420B2010B0460C03EAB02E0C0204B06A0C03A4B0200C0001BD0607F2E0F 0009
A02D4C0000B0981BD090B0982B0282B0041B1105B0282B005AB1502B02227F30DF 0010
A02EAB0020B06C2BDC02B0601B18F2B04C0B0201B0002B0202C0000B04207F2F5F 0011
A0300B2010B0460C03EAB02E0C0204B0201B0003B06A0C0396BC1A0B834A7F2E5F 0012
A0316BD220B8300B0988B0288B0005B1319BD020B8343B0980B0641B04207F2FDF 0013
A032CB200CB06A0C03B4B0581B0420B200CB0601B0420B2008B06A0C03BE7F2D2F 0014
```

```
A0342B0581B0420B2008B0580B0601B0606B18EDB0460C03EABC0C6BD0207F2C3F   0015
A0358B8343B0980B0201B0001B0202C0000BD4A0C0202B0420B2014B05817F32DF   0016
A036EB0202C0101BD4A0C0202B0420B2014B0601B0420B2010B0581B02027F330F   0017
A0384C0000B0420B2010B0580B0601B0603B18E7B0460C03EAB04C0B04207F306F   0018
A039AB200CB0420B2018B12B8B045BB04C0B0201B0001B0202C0000B04207F311F   0019
A03B0B2014B045BB0203B834AB0204C0001B1004B0203C0001B0204B834A7F31CF   0020
A03C6B0205B0004BCD33B0605B16FDB045BBD820C0202C0000B0200B1E007F2CBF   0021
A03DCB0420B2034B04C0B0201B0003B0420B2010BD820C0202C0000BD8207F311F   0022
A03F2C0202C0101BD820C0203B837CB02E0B83E0B0460B00707F502F             0023
50224LEFT$ 50254RIGHT$50286UPPER$502C6LOWER$50306SWAP  7F2D0F        0024
:       99/4 AS                                                      0025
```

This collection is for all TI99 User. If like it, join on International TI99 forum present on AtariAge.com
An regards and greetings from Italy at all TI people past and future :-)

Astharot

BASIC-ERWEITERUNG für den TI-99/4A EXT. BASIC

Besondere Vorzüge des Programms:

1. Es läuft in TI Extended Basic und braucht keinerlei Erweiterung. Fast alle anderen Utility/User-Programme benötigen die 32K-Erweiterung und/oder ein Diskettenlaufwerk.
2. Es erleichtert das Programmieren auf dem TI-99 sehr. 13 neue Befehle stehen unmittelbar zur Verfügung. Sie eignen sich sowohl für Spiele-Programmierer (z. B. Boing/Crash) als auch für Profis (z. B. Mouse/Change/etc.).
3. Das Anpassen von fremden Programmen an den TI-99 wird erheblich erleichtert. In anderen Basic-Versionen übliche Befehle (z. B. Beep/Get/Left/Right/etc.) werden auf dem TI-99 simuliert.
4. Natürlich gibt es auch einiges, was sich schon viele TI-User gewünscht haben, z. B. eine Abfrage ob der Speech Synthesizer angeschlossen ist und eine Möglichkeit, das Programm vom Programm selbst ausgehend zu löschen (Verwendung z. B. als Listschutz-Falle).
5. Trotz dieser vielfältigen Möglichkeiten der BASIC-ERWEITERUNG bleiben für das eigene Programm noch rund 12 KB frei.

Das Programm BASIC-ERWEITERUNG stellt somit eine große Hilfe für alle TI-User dar.

Die 13 Befehle werden mit „CALL Name" aufgerufen und wie die normalen, eingebauten CALL-Befehle behandelt. Im einzelnen können Sie folgende Befehle verwenden:
1. ***CALL BEEP*** erzeugt einen „Piep-Ton" wie bei Input oder Accept
2. ***CALL CL*** löscht den Bildschirm, aber nicht wie Cal Clear sofort, sondern langsam und „echt computermäßig" Zeichen für Zeichen einzeln
3. **CALL CRASH** erzeugt einen „Farb-Blitz" und ein typisches Explosionsrauschen (Verwendung vor allem oft in Spielen)
4. **CALL BOING(A)** erzeugt ein Gong-ähnliches Geräusch mi

der Grundfrequenz A

5. **CALL SPEECH(A)** wenn der Sprachsynthesizer angeschlossen ist, wird A gleich eins gesetzt, wenn nicht, wird A gleich null gesetzt

6. **CALL RAN(UG,OG,A)** der Variablen A wird eine Zufallszahl zwischen der Untergrenze UG und der Obergrenze OG zugeordnet

7. **CALL NEW ACHTUNG!!** Dieser Befehl simuliert NEW innerhalb eines Programms. Nach der Durchführung von CALL NEW wird das gesamte Programm unwiederbringlich gelöscht. Der Speicher ist wieder frei.

8. **CALL CHANGE(A$,B$)** Dieser Befehl ordnet der Variablen A$ den String B$ zu und der Variablen B$ den String A$ (Tausch, Verwendung vor allem in Sortier-Programmen)

9. **CALL WAIT(A)** Hält das Programm an und fährt erst fort, sobald die Taste mit dem ASCII Code A gedrückt wurde. Gibt man für A null ein, so reicht der Druck auf eine beliebige Taste.

```
30000 !=====================          BASIC ERWEITERUNG          ==================
====          TI99-4A EXT.BASIC          ======================
30010 !                    (C)1985 PATRICK SCHMITZ                AM ZEHNTHOF 4
          5480 REM.-UNKELBACH
30020 !
30030 SUB BEEP :: CALL SOUND(-111,1400,2):: SUBEND :: SUB WAIT(A)
30040 CALL KEY(0,K,S):: IF S<>1 THEN 30040 ELSE IF K=A OR A=0 THEN SUBEXIT ELSE
30040
30050 SUBEND :: SUB GET(A$)
30060 CALL KEY(0,K,S):: IF S<>1 THEN 30060 ELSE A$=CHR$(K):: SUBEXIT ELSE 30060
30070 SUBEND :: SUB SPEECH(A):: CALL PEEK(-28672,A):: IF A=96 THEN A=1 ELSE A=0
30080 SUBEND :: SUB RAN(X1,X2,A):: RANDOMIZE :: A=INT(RND*(X2-X1+1))+X1 :: SUBEN
D :: SUB NEW :: ON ERROR 30090 :: RUN ""
30090 END :: SUBEND :: SUB CHANGE(A$,B$):: X$=A$ :: A$=B$ :: B$=X$ :: SUBEND ::
SUB LEFT(A$,B$,A):: B$="" :: FOR Z=1 TO A
30100 B$=B$&SEG$(A$,Z,1):: IF Z=LEN(A$)THEN 30120
30110 NEXT Z :: SUBEXIT
30120 SUBEND :: SUB RIGHT(A$,B$,A):: B$="" :: FOR Z=1 TO A
30130 IF LEN(A$)<Z THEN 30140 ELSE B$=SEG$(A$,LEN(A$)+1-Z,1)&B$
30140 NEXT Z :: SUBEND :: SUB BOING(A):: IF A<110 THEN A=110 ELSE IF A>44733 THE
N A=44733
30150 FOR Z=1 TO 14 STEP 3 :: CALL SOUND(-100,A,Z,A+2,Z+8,A+4,Z+12):: NEXT Z ::
SUBEND :: SUB CRASH :: CALL SCREEN(16)
30160 CALL SCREEN(12):: CALL SCREEN(7):: CALL SCREEN(8):: FOR Z1=1 TO 6 STEP 2 :
: FOR Z=-5 TO -7 STEP -1 :: CALL SOUND(-200,Z,Z1):: NEXT Z :: NEXT Z1 :: SUBEND
30170 SUB CL :: FOR Z1=1 TO 24 :: FOR Z=1 TO 32 :: CALL HCHAR(Z1,Z,30):: CALL HC
HAR(Z1,Z,32):: NEXT Z :: NEXT Z1 :: SUBEND
30180 SUB MOUSE(A$):: CALL SPRITE(#1,43,16,1,17):: ZE=1 :: SP=3
30190 CALL KEY(1,K,S):: IF K<>18 THEN 30200 ELSE CALL BEEP :: CALL GCHAR(ZE,SP,A
):: A$=CHR$(A):: SUBEXIT
30200 CALL JOYST(1,X,Y):: X=SGN(X):: Y=-SGN(Y):: IF ZE+Y<1 OR ZE+Y>24 OR SP+X<1
OR SP+X>32 THEN 30190 ELSE ZE=ZE+Y :: SP=SP+X
30210 CALL LOCATE(#1,ZE*8-7,SP*8-7):: GOTO 30190 :: SUBEND
```

Tips und Tricks für den TI-99/4A

1. CALL LOAD (-31888,63,255) Entspricht CALL FILES(Ø) und legt das Disketten-Betriebssystem lahm.

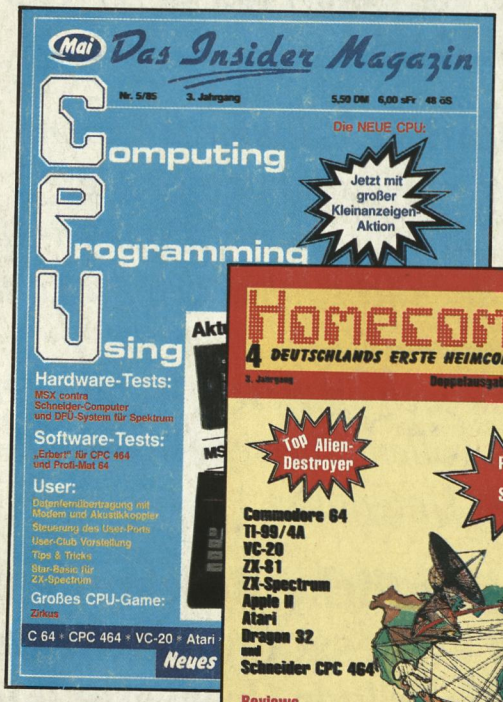
2. CALL LOAD(-31788,232) Schaltet in den Multicolormodus um.

3. CALL LOAD(-318Ø4,Ø,36) Entspricht dem BYE-Befehl.

4. CALL PEEK(-28672,A) Mit diesem Befehl kann man prüfen, ob der Sprachsynthesizer angeschlossen ist.
A=Ø Sprachsynthesizer nicht angeschlossen
A=96 Sprachsynthesizer ist angeschlossen

von
Holger Kusch

Ihr Erfolgsprogramm – aus dem Tronic-Verlag

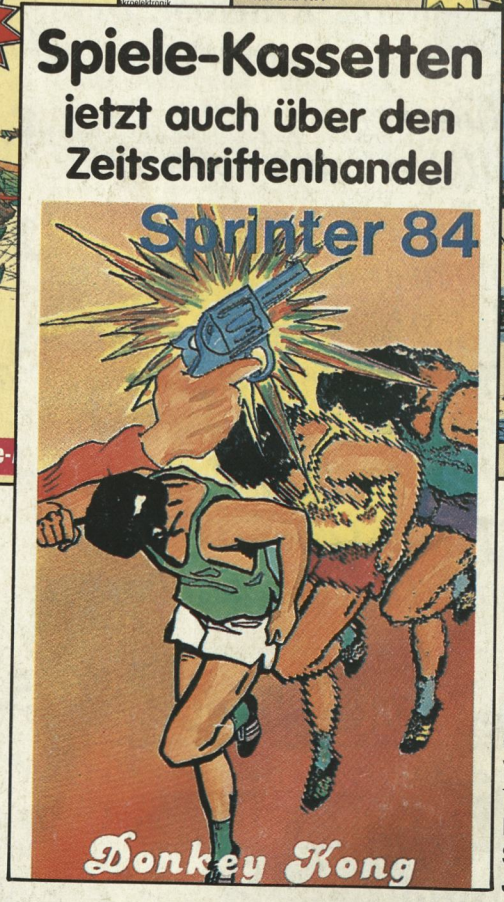


CPU
zweimonatlich
Format 21 x 28 cm
6,– DM

Compute mit
monatlich
Format 21 x 28 cm
2,80 DM

Homecomputer
zweimonatlich
Format 21 x 28 cm
6,– DM

Computronic
zweimonatlich
Format 21 x 28 cm
6,50 DM



Spiele-Kassetten:
Commodore 64
VC-20, Atari,
TI-99, ZX-Spectrum,
Laser 2001,
Schneider CPC 464
Je 19,90 DM

Spitzenprodukte

für den wachsenden Markt der Heimcomputer-Anwender

Informationen: Tronic-Verlag, Landstraße 29

Postfach 41, 3444 Wehretal 1, Ruf (0 56 51) 4 06 93
Vertrieb: Verlagsunion, Friedrich-Bergius-Straße 20, 6200 Wiesbaden, Ruf (0 61 21) 26 60